Smartwatch-Based Keystroke Inference Attacks and Context-Aware Protection Mechanisms

Anindya Maiti[X], Oscar Armbruster[X], Murtuza Jadliwala[X], and Jibo He[O]
[X]Electrical Engineering and Computer Science Department
[O]Psychology Department
Wichita State University, USA
{axmaiti, oxarmbruster, murtuza.jadliwala, jibo.he}@wichita.edu

ABSTRACT

Wearable devices, such as smartwatches, are furnished with state-of-the-art sensors that enable a range of context-aware applications. However, malicious applications can misuse these sensors, if access is left unaudited. In this paper, we demonstrate how applications that have access to motion or inertial sensor data on a modern smartwatch can recover text typed on an external QWERTY keyboard. Due to the distinct nature of the perceptible motion sensor data, earlier research efforts on emanation based keystroke inference attacks are not readily applicable in this scenario. The proposed novel attack framework characterizes wrist movements (captured by the inertial sensors of the smartwatch worn on the wrist) observed during typing, based on the relative physical position of keys and the direction of transition between pairs of keys. Eavesdropped keystroke characteristics are then matched to candidate words in a dictionary. Multiple evaluations show that our keystroke inference framework has an alarmingly high classification accuracy and word recovery rate. With the information recovered from the wrist movements perceptible by a smartwatch, we exemplify the risks associated with unaudited access to seemingly innocuous sensors (e.g., accelerometers and gyroscopes) of wearable devices. As part of our efforts towards preventing such side-channel attacks, we also develop and evaluate a novel context-aware protection framework which can be used to automatically disable (or downgrade) access to motion sensors, whenever typing activity is detected.

CCS Concepts

•Security and privacy → Privacy-preserving protocols; *Side-channel analysis and countermeasures;* **•Human-centered computing → Ubiquitous and mobile computing;**

Keywords

Smartwatch, keystroke, sensor, wearable, privacy.

ASIA CCS '16, May 30-June 03, 2016, Xi'an, China

DOI: http://dx.doi.org/10.1145/2897845.2897905

1. INTRODUCTION

Technologies that fueled the rapid growth of modern electronic computers are also catalyzing development in wearable computing. Most modern wearable devices, such as smartwatches, are capable of a range of context-aware applications, including personal assistance, health and wellness monitoring, personal safety, and corporate solutions, to name a few. Behind the scenes, a wide range of highly precise sensors provide the contextual information required by these applications to provide a seamless and personalized user experience.

Unfortunately, applications with malicious intents can also covertly use these sensors to collect private information, without user-consent. Many of the side-channel attacks implemented for smartphones can be readily applied to smartwatches. For example, a malicious application with appropriate hardware support can stealthily capture photos or video of the user and their surroundings with camera[11], record ambient sounds with microphone [21], and track and predict location activities using GPS [18]. As modern smartphone operating systems recognized open access to these sensors as apparent privacy risk, applications' permission to access these sensors were made user-manageable. Additionally, use or access to certain sensors also include explicit user-notifications, for example, a notification icon appears on Android and iOS when the GPS sensor is being accessed by an application. Yet, malicious applications found novel side-channel attacks to infer private information through indirect means. For example, gyroscope, a sensor usually used to detect changes in device orientation, can potentially be used to detect and recover audio speeches [17]. Another example of a distressing side-channel attack is inferring keystrokes using emanations captured by motion sensors such as accelerometer and gyroscope [16, 19].

Although many of the side-channel attacks designed for smartphones are not directly applicable in wearable devices such as smartwatches, we anticipate that innovative attacks can hugely benefit by the way these wearable devices are used. In practice, smartphone usage is highly intermittent (for example, it has been shown that on an average a smartphone is used only 58 minutes per day [2]), and these devices spend a majority of their time in a constrained (e.g., in users' pockets) or in an activity-less (e.g., on a table) setting where most on-board sensors are partially (or completely) non-functional, thereby limiting the inferential capabilities of malicious applications that attempt to take advantage of data from these sensors. On the contrary, wearable device

usage is much more persistent as they are always carried by users on their body in the same natural position as their traditional counterparts and users are much more naturally habituated to these devices. Our hypothesis is that, as wearable smart devices are persistently and uniquely used, sensors on them are able to capture a continuous stream of user-specific contextual data, access to which if not controlled appropriately, can be potentially exploited by malicious applications to infer sensitive user information.

In this paper, we show that unaudited access to motion sensors featured on most smartwatches can inadvertently lead to significant leakage of information relating to users and their surrounding. We demonstrate that a malicious application, with access to motion sensor readings of a smartwatch, can decode the keystrokes made on a QWERTY keyboard while wearing the smartwatch on one hand. We achieve this based on the observed relative physical position of keystrokes and direction of transition between pairs of keystrokes. We then recover the typed words by mapping the captured 'motion' of each word to pre-formed motion profiles of words in an English language dictionary. Due to the distinctive nature of perceptible sensor data on smartwatches, straightforward adaptation of earlier side-channel keystrokes attacks based on emanation of electromagnetic, acoustic or vibration pulses generated by a keystroke, is not befitting. A comprehensive empirical evaluation of our keystroke inference framework show significantly high word recovery rates.

As evident from our experimental results, the threat to privacy posed by side-channel attacks using wearable devices is substantial. However, there have been very limited efforts from the research community to effectively defend against such side-channel attacks in a user-friendly fashion. We propose and implement a new context-aware protection framework which can automatically activate various protection mechanisms whenever typing activity is detected. We also empirically evaluate the protection framework in real-life usage scenario.

2. RELATED WORK

Emanation based side-channel inference attacks date back to the World War II era [12]. The primary types of emanations include electromagnetic signals, sounds, and vibrations. Previous studies demonstrated the use of electromagnetic emanation to eavesdrop on contents displayed on a CRT or LCD screen [24, 14] from a distance and with opaque obstacles in between. Similar attacks using electromagnetic emanations have also been shown to work against CPU chips [3], smart cards [20], data carrying cables [22], and keyboards (wired or wireless) [25]. Optical emanation, contained in the band of electromagnetic spectrum perceptible to human eyes, present a different form of leakage for display devices. The light released from display devices may reflect off various surfaces in front of the screen, and reach an eavesdropper. Successful reconstruction of the displayed information has been demonstrated based on reflection such as from walls [13], shiny objects [7], and even from viewer's eyes [5]. While electromagnetic emanation based attacks are certainly effective, the need of specialized equipment and it's concealed placement near to the target poses difficulty. Similarly for side-channel attacks based on optical emanations, the eavesdropping equipment must be placed in line of sight of the target.

Side-channel attacks based on acoustic or sound emanations are much more feasible because of the popularity of personal devices featuring microphones. Microphones are inexpensive, and can be easily concealed because of their compact form factor. Furthermore, if a target's microphone enabled device (such as smartphones, tablets, etc.) is hijacked, it can act as a disguised eavesdropping equipment. As much as 90% of English text printed by a dot-matrix printer can be successfully recovered, by learning the acoustic emanations released by the printer [6]. The other major use of acoustic emanation has been in keystroke inference attacks, which targets to recover key presses on a nearby computer keyboard [4, 9]. Similar keystroke inference attacks can be carried out using surface vibration emanation generated during keystrokes [16, 8]. Vibrations of nearby surfaces caused by human voice can also be recorded, and used to decode speeches [17]. While systems to record vibrations may be difficult to conceal, Marquardt et al. [16] proposed the use of a smartphone's accelerometer to record vibrations near keyboards. If an adversary is able to infect their target's smartphone with a malicious application which can record and transmit sensor data stealthily, it can serve as a very effective eavesdropping tool.

However, a critical requirement of learning based side-channel attacks using electromagnetic, acoustic, or vibration emanation, is that the target and eavesdropping equipment must not be disturbed. Change in either's position or orientation will render the training data futile, thereby making recovery of target information impossible. This also means that training must be performed in the same setting as the attack, which may not always be feasible. For example, in case of [16], if the target person puts his/her smartphone one day on the left side of the keyboard and another day on the right side, the vibrations captured by the accelerometer will be significantly different, resulting in failed recovery of typed text. Our attack setting, which uses motion data from a wearable device to infer keystrokes, is largely unaffected due to similar constraints as most people wear and use these devices in a very standard fashion (for example, smartwatches are almost always worn on the left wrist by most people). Moreover, our attack mechanism and wrist motion characterization framework is very general and can be easily extended to work in scenarios comprising of non-traditional usage of these wearable devices (for example, users wearing the watch on the right hand instead).

During the final phase of completing this work, we came across recently published works which demonstrate the ability to infer keystrokes using smartwatch. Maiti et al. [15] used machine learning to train classifiers based on the slight differences in wrist movements observed while tapping numeric keys on a handheld smartphone keypad, depending on the location of the key on the screen. The trained classifiers are then used on test data to perform multiclass classification between the ten keys. Similar to our work, Wang et al. [26] demonstrate the feasibility of keystroke inference attack using a smartwatch, on a QWERTY keyboard. However, their attack framework is very different from ours. We also conduct a comprehensive evaluation of our attack framework and preliminary results indicate that our approach leads to better inference accuracy compared to [26]. However, [26] has a different experimental setup, due to which we are unable to make a comprehensive comparison like we do with [16] and [9].

Interestingly, none of the recent works on side-channel keystroke inference attacks propose or implement a practical protection mechanism. Some of the previous work using smartphone sensors as side-channels, briefly suggest operating system developers to provide users with fine-grained control over application's permissions to every sensors [10]. But without knowing which application is malicious, the user may have to toggle sensor access back and forth for all the installed applications. Other research efforts vaguely suggest to restrict the precision at which applications are allowed to access the sensors [19, 17, 15]. However, regulating sensor precision will result in poor application performance, for example, gaming applications will have slow controls and response, mapping applications will be delayed/inaccurate, etc. Moreover, some sensors (such as camera and microphone) will be rendered unusable at very low sampling rates. In the more recent work using smartwatches as a side-channel [26], Wang et al. completely overlooked the necessity of having protection mechanisms. In this paper, we not only demonstrate the feasibility of keystroke inference attacks using smartwatches as a side-channel, but we also design, implement, and evaluate a new context-aware protection framework to defend against such attacks.

3. ATTACK DESCRIPTION

In this paper, we demonstrate the feasibility of a keystroke inference attack against a user typing on an external QWERTY keyboard by using smartwatch motion sensors. Because of limitations faced by emanation-based keystroke inference attacks, and multiple technical challenges in implementing them on a smartwatch, we pursue a slightly different approach for our attack where we focus on capturing and using keystrokes related wrist motion or movement characteristics.

We observed that the wrist movements made while typing a fixed sequence of letters on a keyboard are highly similar and consistent across multiple trials involving a single typer. This gave us the intuition that an adversary can create a dictionary of commonly used words (words are nothing but fixed sequence of letters), along with their corresponding wrist movement patterns. During the attack, the adversary can simply match the eavesdropped wrist movement pattern to the closest matching pattern in the dictionary. Intuitively, the recovery can be highly accurate if the dictionary is carefully created and comprises of all words that the target is expected to type. However, the recovery rate also depends on how the wrist movement patterns are characterized (which we will explain in Section 4.1) and the granularity of the captured wrist movement data (which is generally limited by the eavesdropping sensor's maximum sampling frequency).

For carrying out the proposed inference attack, an adversary requires an eavesdropping device that is capable of continuously recording wrist movements, while avoiding detection. A modern commercial-off-the-shelf smartwatch, which is generally equipped with a range of sensors (especially motion sensors), can easily serve as such an eavesdropping device. Other forms of wrist wearable devices such as activity trackers and fitness bands, are typically also equipped with motion or inertial sensors, and can also be used as an eavesdropping device for the proposed attack. In this work, without loss of generality, let's assume that the adversary exploits the smartwatch as an eavesdropping device. However, a bigger challenge is how does an adversary gain access to the motion data captured on the smartwatch. This can be achieved by an adversary installing a malicious application that has access to the motion sensors on the target's smartwatch such that the application is able to stealthily capture and transfer the captured motion data to the adversary.

This is feasible because, even though an application's access to the some sensors (e.g., GPS and camera) is generally user-managed or restricted on most modern mobile operating systems such as Android and iOS, access to motion sensors (such as, accelerometer and gyroscope) remains highly unregulated. An adversary can easily install the malicious application on the target smartwatch by various means, for example, by gaining physical access to the device or through social engineering (e.g. masquerading as a legitimate application, pretexting, baiting, phishing etc.). The malicious application can then stealthily collect and transfer motion data by masquerading as, or piggy backing on, useful application data and network traffic. In other words, the infected smartwatch now acts as an eavesdropping device that the targets' themselves place on their wrist, and unsuspectingly have it on their wrist while typing on a keyboard, as depicted in Figure 1.

The malicious adversarial application on the smartwatch records the linear accelerometer data (linear accelerometer measures the acceleration experience by the device, excluding the force of gravity) and microphone data. In the proposed attack, the acoustic data recorded by the microphone is not used for keystroke inference, but rather just to identify keystroke events (as explained in detail in Section 4.2.2). Due to the impracticality of an on-screen keyboard on the small smartwatch screens, an adversarial smartwatch application can seek access to the microphone in order to support voice commands or dictation, which is common. Alternatively, keystroke events can also be recognized by solely using the motion sensors, as accomplished in Marquardt et al. [16]. As mentioned earlier, the recorded sensor data is then transmitted by the malicious application to the adversary directly over the Internet by masquerading as useful communication or by piggyback on communications from other applications. In an effort to save battery power (necessary for avoiding detection), the recording and communication process may be initiated remotely by the adversary or based on periodic activity tracking.

Figure 1: An exemplary setup where a person is typing on a QWERTY keyboard, while wearing a Samsumg Gear Live smartwatch on left hand. A similar setup is used in our experiments.

4. THE ATTACK FRAMEWORK

In this section, we present our model for identifying keypress events from raw motion sensor data. We then discuss our attack framework, and an experimental setup for evaluating the framework.

4.1 Modeling Key Press Events

With the maximum supported linear accelerometer sampling rate (∼50-70Hz) being much lower than that of smartphones (∼200-300Hz), the difficulty in recognizing individual keys is greatly increased when using a smartwatch. To overcome this shortcoming, we attempt to identify *pairs* of key presses or keystrokes by learning the relationship between them. While typing a word, there will be one key press for each character or letter in the word. Let K_i, K_j be two consecutive key press events, signifying two consecutive characters or letters of a word. We characterize the relation, $rel(K_i, K_j)$, between any two consecutive key press events K_i, K_j as follows:

- **Horizontal Position:** The location $loc(K_i)$ of each keystroke event relative to a 'central-line' dividing the keyboard into left (L) and right (R) halves. *The rationale behind this classification is that the wrist movement will be more pronounced for typing a key on the same side as the watch-wearing hand.*

- **Transitional Direction Between Consecutive Key Presses:** The direction $dir(K_i, K_j)$ represents the direction of wrist movement between consecutive key presses K_i and K_j on watch-wearing side of the keyboard. The possible directions (or values for $dir(K_i, K_j)$) are N, E, S, and W, representing geographical north, east, south, and west, movement respectively. An additional classification is O, if $K_i = K_j$. *The rationale behind this classification is that the direction of transition between a pair of keystrokes will be reflected in the wrist movement.*

With the above classification, the relationship between two consecutive key press events is defined as follows:

- When either K_i, K_j, or both, occur on the non-watch wearing side of the keyboard, $rel(K_i, K_j) = loc(K_i) \mid\mid X \mid\mid loc(K_j)$, where '$X$' implies that direction cannot be determined. The intuition behind such an assignment is that it is not possible to determine the direction of transition when at least one of the pressed key is not on the watch-wearing side of the keyboard.

- When both K_i and K_j occur on the watch-wearing side of the keyboard, $rel(K_i, K_j) = loc(K_i) \mid\mid dir(K_i, K_j) \mid\mid loc(K_j)$.

A *word-profile* for a word can then be derived by concatenating the relation between every consecutive pair of letters in the word. For example, the word "boards" can be broken down in to five pairs of keystrokes {bo, oa, ar, rd, ds}, i.e., word-profile for the word "boards" is $rel(bo).rel(oa).rel(ar).rel(rd).rel(ds)$. With the setup for a QWERTY keyboard, as shown in Figure 2, and the entire L/R and N/E/S/W/O classification listed in Table 1, the word-profile of "boards" will be:

RXR . RXL . LEL . LSL . LWL

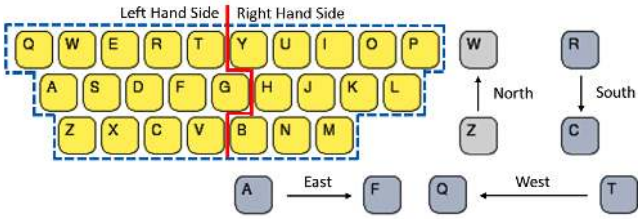


Figure 2: The keyboard is divided in to left (L) and right (R) halves, shown by the solid red line. Examples of N, E, S, and W classification are also shown. Each direction has 90° field of view from center of the key. Keys that fall on the boundary are categorized in the direction where greater area of the key lies.

Table 1: L/R classification of individual keys and N/E/S/W/O classification of character-pairs, assuming smartwatch is worn on left hand.

L	q, w, e, r, t, a, s, d, f, g, z, x, c, v
R	y, u, i, o, p, h, j, k, l, b, n, m
N	aq, aw, sw, se, de, dr, fr, ft, gt, zq, zw, ze, za, zs, xw, xe, xr, xs, xd, ce, cr, ct, cd, cf, vr, vt, vf, vg
E	qw, qe, qr, qt, qs, qd, qf, qg, qx, qc, qv, we, wr, wt, wd, wf, wg, wc, wv, er, et, ef, eg, ev, rt, rg, ae, ar, at, as, ad, af, ag, ax, ac, av, sr, st, sd, sf, sg, sc, sv, dt, df, dg, dv, fg, zr, zt, zd, zf, zg, zx, zc, zv, xt, xf, xg, xc, xv, cg, cv
S	qa, qz, wa, ws, wz, wx, es, ed, ez, ex, ec, rd, rf, rx, rc, rv, tf, tg, tc, tv, az, sz, sx, dx, dc, fc, fv, gv
W	wq, eq, ew, ea, rq, rw, re, ra, rs, rz, tq, tw, te, tr, ta, ts, td, tz, tx, sq, sa, dq, dw, da, ds, dz, fq, fw, fe, fa, fs, fd, fz, fx, gq, gw, ge, gr, ga, gs, gd, gf, gz, gx, gc, xq, xa, xz, cq, cw, ca, cs, cz, cx, vq, vw, ve, va, vs, vd, vz, vx, vc
O	qq, ww, ee, rr, tt, aa, ss, dd, ff, gg, zz, xx, cc, vv

The main idea behind our attack is that the adversary will have a pre-processed dictionary of well-known (or targeted) words and their corresponding word-profiles (formed as discussed before). These word-profiles are used in distinguishing between candidate words from the dictionary. Given the motion data, the adversary will attempt to infer word-profiles from the motion data and then use the pre-processed dictionary to determine the typed word by comparing the inferred word-profile to the word-profile in the dictionary. However if the dictionary is large, more than one word may have the same word-profile. Such *collisions* may result in incorrect predictions, and thus, reduce the accuracy of the inference attack by the adversary. In such cases, a frequency-based selection (as discussed in Section 5) could yield better word recovery results. Similarly, defining word-profiles by using additional fine-grained directional data (e.g., NE, SW, etc.) could reduce the number of collisions and improve inference accuracy, however it will also increase the attack execution time for the adversary.

4.2 Keystroke Inference Attack

Broadly, our proposed inference attack comprises of a *learning phase* (Figure 3) that is followed by the *attack*

phase (Figure 4). However, before initiating the learning phase, the adversary must define the classification parameters. This includes deciding the keys in L and R halves, determining the hand on which the smartwatch is worn, and accordingly form all perceptible transitions. In our experiments, we suppose that the target is wearing the smartwatch on his/her left hand and the keyboard is divided in L and R halves as shown in Figure 2. Accordingly, all 196 possible transitions with the watch-wearing hand are listed in Table 1. However, the proposed attack could easily be modified (with little effort) for the watch worn on the right hand or for other forms of L/R division of the keyboard. After these parameters are determined, the learning phase can begin.

4.2.1 Learning Phase

The purpose of this phase is to construct trained classification and prediction models for use during the attack phase. Training of these models comprises of the following four steps: (i) *data collection*, (ii) *feature extraction*, (iii) *word labeling*, and (iv) *supervised learning*. To ensure uniformity in the learning models, the training data is chosen such that it has equally distributed features. This can be achieved by using a large set of randomly generated words, uniformly covering all keys and apprehensible transitions.

Data Collection: There are two types of data recorded by our custom Android Wear attack (or data collection) application. First, is the motion data just before and immediately after a keystroke. Second, is the entire transition data between two keystrokes that occur on the watch-wearing side of the keyboard. Both types of recorded data are the *linear accelerations* experienced by the smartwatch, as sensed by it's linear accelerometer sensor. The sampled linear accelerometer readings are composed of instantaneous three dimensional linear acceleration along the X, Y, and Z axes. One of the authors (pretending to be the adversary) typed a set of 1000 random English words which uniformly covered all 26 keys and 196 transitions, without any fixed ordering or timing. Note that the number of possible transitions will be 144 if the target wears the smartwatch on the right hand and the keyboard is divided into the same L and R halves. The data collection application also clocks and tags the ground truth of the typed keys, which helps simplify the feature extraction and labeling process later, which in turn, ensures error-free training.

Feature Extraction: Feature extraction aids in dimensionality reduction by eliminating redundant measurements. For (L/R) keystrokes we compute a comprehensive set of 24 type of features such as mean, median, variance, standard deviation, skewness (measure of any asymmetry) and kurtosis (to measure any peakedness). We use multiple inter and intra-axis time domain features to capture the correlation between movement on the three axis, and frequency domain features to identify the different rebounding (or oscillatory) motion of the wrist. However, in case of (N/E/S/W/O) labeling, we observed that the transition period was varying widely based on typing speed and word composition. As a result, it is impossible to represent the entire transition in a fixed length time-domain feature vector (as used in feature vectors with (L/R) labels). As a solution, we use frequency-domain features such as Fast Fourier Transformation (FFT) of the transition data.

Labeling: Each training word is broken down into its constituent characters and character-pairs. As a result, a word of length n letters would be broken into n characters and $n-1$ character-pairs. Feature vector of each keystroke is labeled (L/R) using the ground truth characters recorded during data collection. Feature vectors of directions are labeled (N/E/S/W/O) by calculating the direction between character-pairs obtained from the same ground truth. An additional processing is performed to select a set of character-pairs with even distribution of L and R and N, E, S, W and O labels. Note that the number of N, E, S, W and O labels will be approximately one-fourth of the number of L _ L, L _ R, R _ L and R _ R pairs because the direction is determined only in case of L _ L transition (or R _ R if the target wears the smartwatch on right hand). For L _ R, R _ L and R _ R character-pairs, the direction for transition cannot be determined (which is denoted by X in the word-profile), and thus, they are not used in the training phase.

Supervised Learning: We created two separate training models that will be used during the attack phase to classify keystrokes and keystroke-pairs. The two trained models are *L-R* and *N-E-S-W-O* neural networks for classifying (L/R) and (N/E/S/W/O) feature vectors, respectively. Because of the complex interactions possible between consecutive keystrokes, we train our classifiers using *neural networks*. Neural networks are specifically useful in discovering these complex interactions between the corresponding feature vectors, and improving the classification model based on it. Our L-R neural network uses a *back-propagation algorithm* for learning at a rate of 0.01 and with a momentum of 0.99. This neural network has 30 hidden layers and training was performed for 2000 epochs. Our N-E-S-W-O neural network also uses a back-propagation algorithm for learning at a rate of 0.001 and with a momentum of 0.99. This neural network has 100 hidden layers and the training was performed for 1000 epochs. These parameters for our neural network based classifiers were chosen heuristically. Training of these neural networks completes the learning phase.

4.2.2 Attack Phase

The attack phase follows a similar procedure as the learning phase, with the exception that the goal here is to recover test words using the trained neural networks-based classification models from the learning phase. In order to do so, the adversary must first create a dictionary of words, and their corresponding word-profiles, that the target is most likely of typing. The dictionary size can vary from a few words to thousands of words, depending on the target and his/her context. If the adversary is unaware of the target's context, he could also create a large dictionary of most popular or all words in the English language. The dictionary creation involves a preprocessing step to obtain equivalent word-profiles of each word in the L/R and X/N/E/S/W/O representation (as discussed before). The attack phase is then executed in sequential steps of: (i) *data collection*, (ii) *feature extraction*, (iii) *keystroke classification*, and (iv) *word matching*.

Data Collection: The same properties and operations from the data collection operation of the learning phase also applies to the data collection during the attack phase. The only exception is that the malicious Android Wear application does not have the ability to clock and tag ground truth characters. As the smartwatch can only detect motion cause by one (watch-wearing) hand, a significant challenge of the

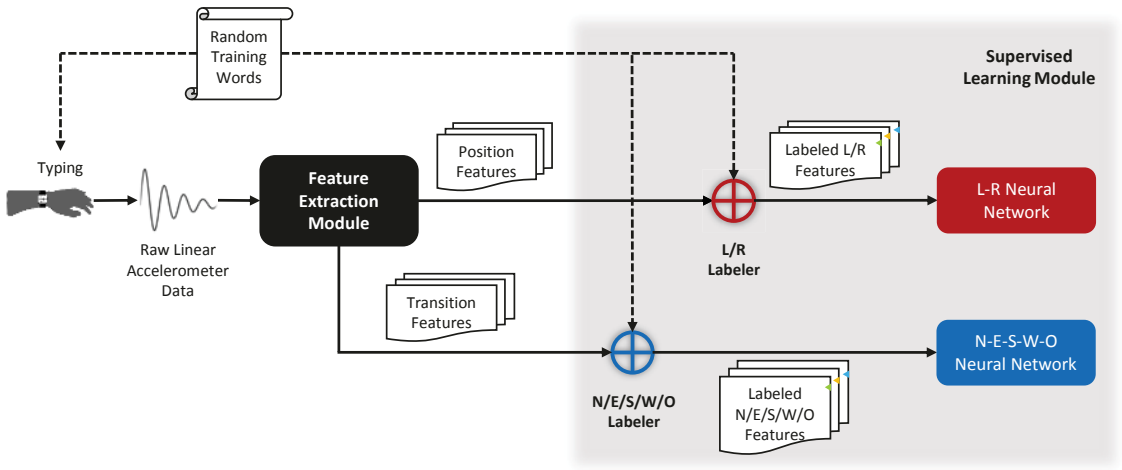


Figure 3: Learning Phase: A high level overview of the data processing architecture used to train the neural networks.

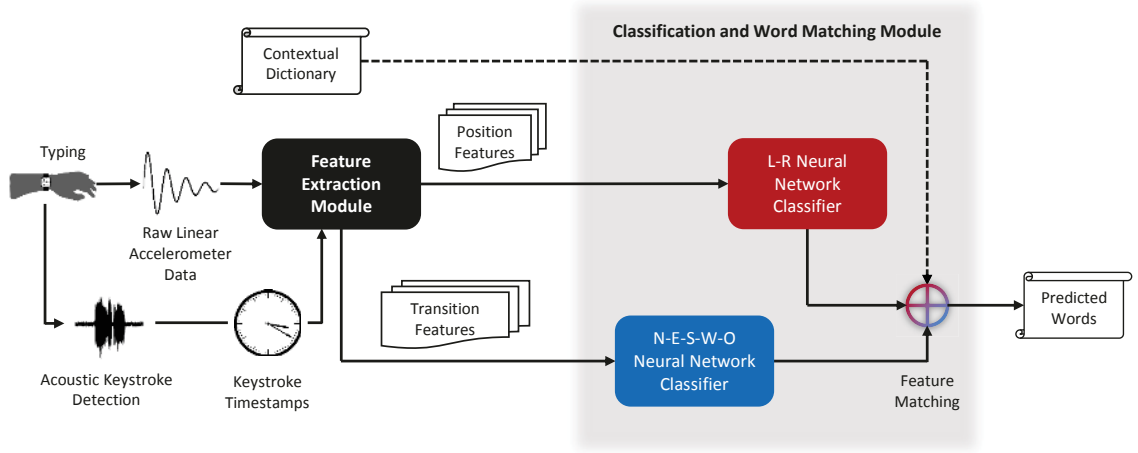


Figure 4: Attack Phase: A high level overview of the data processing architecture used to analyze keyboard input using the trained neural networks.

proposed inference attack is due to the inability to detect keystrokes made by the non-watch-wearing hand. However, our attack framework requires to know the number of typed characters. To solve this problem, here we assume that the adversary can employ an alternate source or sensor on the smartwatch that can detect keystroke events typed by either hand. The intention for using such an auxiliary sensor is not to classify the keystrokes using it, but to clock the time when a keystroke occurs in the stream of raw linear acceleration data. A microphone can perfectly serve this purpose. Even though a smartwatch's microphone is not effective for recovering keystrokes (due to the aforementioned reasons), it can certainly be used to detect the occurrence of a keystroke itself, made by either hand. This is where the naturally close positioning of the smartwatch near the keyboard is beneficial. Thus, the malicious application uses the microphone to detect keystroke acoustics, and in the case a keystroke event is detected from the acoustic signal, it logs a keystroke event in the linear accelerometer data stream. Space key press events are also clocked or logged because they act as word separators. Fortunately, as we empirically determined, space keys are easy to identify in an audio recording because of the key's distinctive sound and frequency of use.

Feature Extraction: During the attack phase, the same features as in the learning phase are extracted from the raw linear acceleration data recorded by the malicious application. The feature vectors are then used to create two sets of data, one for classifying L vs. R, and one for classifying N vs. E vs. S vs. W vs. O.

Keystroke Classification: The adversary initiates the classification process after extracting all feature vectors. The trained L-R neural network is used to predict the (L/R) label for each individual keystroke. Only when a L _ L key-pair is detected in the data stream by the L-R neural network, the N-E-S-W-O neural network classification is conducted to predict the transition direction label (N/E/S/W/O). Otherwise, the transition direction is labeled as X. Using the predicted labels (and the recognized space keys as described earlier), a word-profile is constructed for each word in the keystroke stream. All the constructed word-profiles are then passed as input to a word matching algorithm described next.

Word Matching: Word matching is the final step of the attack phase, where each predicted word-profile of length m is matched with all words of length $m + 1$ in the preprocessed dictionary by the adversary. For each matched word

in the dictionary, a *similarity score* is computed based on the number of matching labels between the predicted word-profile and the corresponding word-profile in the dictionary (see details in Algorithm 1). The dictionary word with the highest similarity score is then output as the word corresponding to the predicted word-profile. For some evaluation experiments, we also use a 'similarity list' made of dictionary words with descending order of similarity scores.

Algorithm 1 Word Matching Algorithm

```
similarityScore = 0
for all words of len(m) ∈ dic do
    for pair = 1 to m − 1 do
        for label = 1 to 3 do
            if dic.word.profile[pair][label] = predicted.profile[pair][label] then
                similarityScore++
            end if
        end for
    end for
end for
return similarityScore
```

4.3 Experimental Setup

In our experimental evaluation of the proposed inference attack and keystroke characterization framework, we use a setup similar to the one shown in Figure 1. We recruit 25 participants[1] who wear the smartwatch on their left wrist and type test words on an external QWERTY keyboard. All data recorded by the smartwatch was transferred to a remote server. Both the training and attack phases are executed on this remote server which is assumed to possess enough computational and storage resources in order to carry out these operations. The specifications of important hardware and software components used in our experiments are outlined below:

1. Smartwatch and sensor hardware: We used the *Samsung Gear Live smartwatch* running *Android Wear build 1.1.1.1944630*. The Gear Live is equipped with an *InvenSense ICS-43430* microphone and an *InvenSense MP-92M* 9-axis Gyro + Accelerometer + Compass sensor. The maximum average linear accelerometer sampling rate achieved in our experiments was 50 Hz. Our data collection application can be readily used on any Android Wear smartwatch, which makes the attack framework compatible with a diverse set of smartwatches.

2. Keyboard hardware: We chose to use the *Anker A7726121* bluetooth keyboard because of its generic design. The bluetooth connectivity aided in accurate labeling of sensor data, by allowing us to aggregate recorded sensor data and corresponding typed character on the smartwatch in very close to real time.

3. Signal processing tool: Most of the features are calculated using MatLab 2015a libraries.

4. Supervised machine learning tool: We used *PyBrain v0.31* to train and test the neural networks in our framework. PyBrain is an open-source modular machine learning library for Python, supporting easy integration with underlying environment.

[1]Our experiments have been approved by Wichita State University's Institutional Review Board (IRB).

5. EVALUATION

We first perform two preliminary experiments (involving only one participant) in order to evaluate (i) the base accuracy of the L-R and N-S-E-W-O classifiers by analyzing a set of test sentences from the set of *Harvard sentences* [1] and (ii) the word recovery accuracy of the proposed inference attack strategy by using a dictionary of ten Harvard sentences and attempting to recover each of the same ten sentences as test data. We choose Harvard sentences because they are phonetically-balanced. In the two preliminary experiments we make the assumption of 'perfect typing', i.e. the participant follows our L/R separation. After the preliminary experiments, we conduct more realistic experiments involving all 25 participants, real-life sentences, larger dictionaries, and without the assumption of perfect typing.

5.1 Feature Accuracy

In the first experiment, we examine the base accuracy of both L-R and N-S-E-W-O classifiers in correctly distinguishing between L/R region for individual letters and N/S/E/W/O transition between pairs of letters. We evaluate our trained classifiers using all the ten sentences in List 6 of Harvard sentences. Interestingly, without any typing errors, the L-R classifier was able to correctly identify 100% of the individual key press events as left or right. However, the N-E-S-W-O classifier had two mis-classifications, resulting in 95% accuracy.

5.2 Basic Text Recovery

Our next experiment examines the percentage of text (in terms of words) correctly matched by the word matcher. In this preliminary experimental results, we observed that the overall percentage of words correctly matched noticeably dropped due to mismatched two and three letter words in the analyzed text. The smaller number of features in these words results in several of these 'small' words having the same word-profile, thus causing more collisions during matching. We also observed that most of these words are generally articles and conjunctions (e.g. an, the, and, or), which can be easily interpolated by analyzing the language semantics of the recovered text. As a result, we opted to consider only 'long' words of four letters or more in all final percentages of recovered words in our remaining experiments. The 'short' words are instead denoted with asterisks ("*") in the recovered sentences.

In this experiment, we used the same ten sentences in List 6 of Harvard sentences, as from the first experiment. Among the 48 words of length four or more, only three were erroneous (93.75% successful recovery). Out of the

Typed Text:	**The**	**show**	**was**	**a**	**flop**	**from**	**the**	**very**	**start.**
Recovered:	***	**sums**	***	*	**flop**	**from**	***	**very**	**start.**

Colliding Word-Profiles:
Show: LXR . RXR . RXL , Sums: LXR . RXR . RXL

Figure 5: Sentence 4 from List 6 of Harvard Sentences. The words 'show' and 'sums' have the same word-profile resulting in a collision in the dictionary.

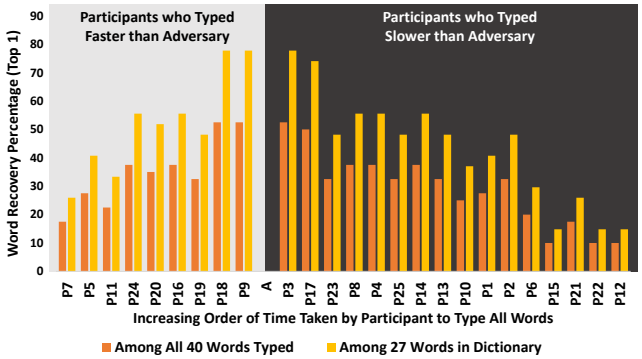


Figure 6: Contextual Dictionary: Percentage of words recovered per participant, presented in descending order of typing speed of the participants.

Figure 7: A comparison of accuracy of our attack with Marquardt et al. [16] and Berger et al. [9]. Note that in spite of not having wrist movement information available from the non-watch-wearing hand, our results are roughly comparable for a very large (60,000 words) dictionary.

three, two had incorrect N/E/S/W/O classification, while the other was due to collisions in the word-profiles. Figure 5 shows the sentence where the collision occurred. The problem of collision will increase with increasing size of the dictionary. However, if we also take in to account second and third ranked similar word-profiles during word matching, this problem can be moderated. Errors in word recovery (especially, due to collisions) can be further diminished by analyzing language semantics, and then selecting the word (from the multiple colliding words in the dictionary) that is semantically best fit.

5.3 Contextual Dictionary

This experiment evaluates how our attack performs when the adversary has some knowledge about what their targets are typing. All the 25 participants typed a paragraph of 40 words (of length four or more) that appear in a *National Public Radio (NPR)* news article on Greece debt crisis, and this experiment simulates eavesdropping on a reporter typing the NPR news article. The dictionary is formed with words that appear in six other news articles related to Greece debt crisis, that were published a week before the target article. The dictionary is also sorted based on frequency of word appearances in the six chosen news articles, which improves our chances of successfully solving a word-profile collision. Figure 6 shows the percentage of words recovered per participant. As one should expect in a real-life attack, out of the 40 words in the target paragraph, only 27 were present in the contextual dictionary. Even so, our framework was able to recover as many as 21 words (for 3 participants), by matching with just the first ranked word in the sorted list of similarity scores (Figure 6). In other words, 21 words were uniquely identified without any ambiguity, for the 3 participants. On the lower end, only 4 words were recovered for 3 participants, but the recovery can be improved by considering words with lower rank in the sorted similarity list. The mean word recovery using only the first ranked word was 31.2% (or 46.2% if we consider only the words present in the dictionary).

5.4 Typing Behavior and Speed

During data collection, we observed that in many instances participants did not follow our assumed layout. Some of the participants frequently used their left hand to press a key on the right side of the keyboard, and vice versa. Upon further investigation we also found that participant who typed slower, were less likely to follow the left and right division of the keyboard. This phenomenon explains why participants who took longer to type all the 40 words saw lesser word recovery rate in Section 5.3 experiments. Figure 6 shows the time taken by the adversary (whose typing was used as the training data) to type the 40 words as A on the horizontal axis. Participants on the right of A typed slower than the adversary, and we can see a trend that the recovery rate drops with slower typing. Interestingly, we see a similar trend on the left of A as well, indicating that recovery rate drops with faster typing. Our speculation is that due to fast typing there may occur overlapping feature regions leading to poorly performing L/R classification, and incorrect L/R classification can significantly affect recovery of words. Combining both the trends we arrive at a conclusion that participants who typed at a similar speed as the adversary were more vulnerable to the attack. For an adversary, the take-home message from this conclusion is that the attack framework can be optimized by training it with a typing speed and style expected from the potential victim(s).

5.5 Comparison to Previous Work

From the above experiments, we saw that relying on exact match with first ranked words may not always result in the best inference accuracy. As pointed out by earlier emanation based keystroke inference attacks [16, 9], more intelligent adversaries may be able to form target sentences with lower ranked words from the sorted similarity list. So, we re-create the experiments conducted by Marquardt et al. [16] and Berger et al. [9] in order to be able to compare our attack framework directly with theirs. We use a similar sized English dictionary of 60,000 words (of length 4 or more), sorted based on frequency of usage in English literature. We reuse 38 of the 40 words typed by participants in Section 5.3 experiment, while remaining 2 (first and last name of former Greek finance minister) are not contained in the 60,000 word dictionary. Figure 7 shows the comparison.

Our attack framework demonstrates comparable accuracies to that of Marquardt et al. and Berger et al. It was able to correctly map test words to the top 10 words in the

sorted similarity list 50.5% of the time, which happen to be significantly higher than the earlier works using smartphone sensors. The word recovery steadily improves as we increase the size of selection from the sorted similarity list, but our attack trails behind the other two in case of very large selections. Note that the complexity of forming sentences with ambiguously recovered words grow exponentially with the selection size. Therefore, achieving a better recovery rate with just top 10 words is more significant than having a better recovery rate using top 500 words. It is also important to remember the distinct challenge faced by our technique where no wrist movement information is available from the non-watch-wearing hand.

We are unable to compare equitably with Wang et al. [26] because of their different experimental setup. However, using a smaller dictionary of only 5,000 words, they were able to narrow down a typed word to 24 possibilities with a 50% chance. In contrast, we use a much larger dictionary of 60,000 words, and our attack is still able to narrow down a typed word to only 25 possibilities with about 52.5% chance.

6. LIMITATIONS

Our proposed movement-based keystroke inference attack using smartwatches circumvents some of the limitations of emanation-based attacks, but it faces new challenges. In this section, we discuss some of them.

- **Ambient Wrist Movement:** In case the target participates in some other activity (for example, having a periodic sip of drink) in between typing, the introduced noise can lead to incorrectly predicted words. However, since each word is treated separately, the error will not propagate.
- **Left and Right Handedness:** Although the same attack framework is applicable independent of the hand on which the smartwatch is worn, classifiers trained using data with the smartwatch worn on the left hand cannot be used to predict words typed while wearing the smartwatch on the other hand, and vice-versa.
- **Inferring Non-Dictionary Text:** Our attack performs well for dictionary words, but is incapable of recovering numeric keys and special characters. As a result, if the adversary is interested in learning data with numbers and/or special characters (such a credit card numbers, strong passwords, etc.), the presented framework and attack will not be directly applicable. However, wrist movements can still be useful in determining approximate position of keys pressed, which may significantly reduce the search space.

7. SMART MITIGATION

Our proposed attack demonstrates the need for reforms on how sensors on smartwatches, and other wearable devices, are accessed by applications. Even innocuous sensors can be used as side-channels to indirectly infer private information. However, there is no straightforward remedy to such privacy threats. In this section, we present a smart countermeasure to prevent such attacks in future.

The simplest way to protect against the presented attack would be to remove the smartwatch from wrist while typing. But repetitive removal of the watch (and remembering when to remove) can become a burden for the user, as a result of

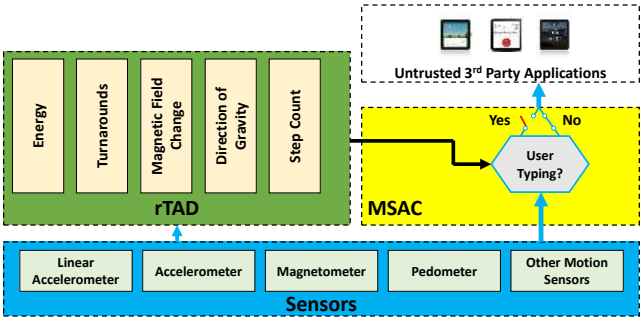


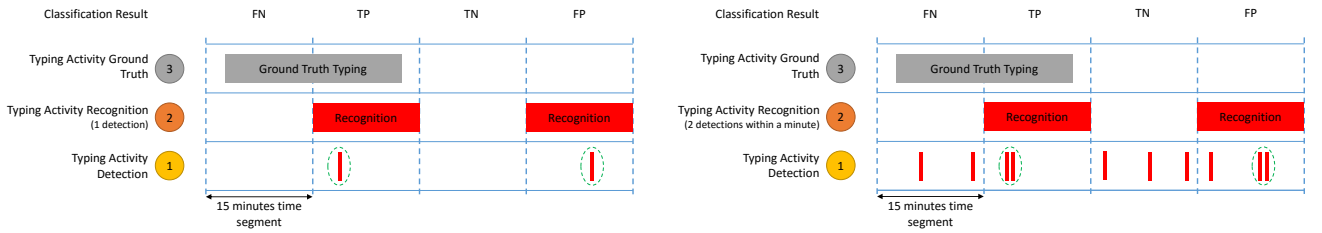
Figure 8: The protection framework against keystroke inference attacks. Third party applications get unrestricted access to motion sensors only when rTAD reports that the user is not typing at the moment.

which, the user may choose to ignore the threat altogether. To draw a favorable balance between utility, usability and privacy while using wearable devices, we need smarter sensor access controls. We feel that sensor access controls need to be context-aware in order to automatically manage an application's sensor permissions, without having the user to manually change these settings repetitively. As part of our efforts to prevent smartwatch based side-channel inference attacks demonstrated earlier in this paper, we design, implement and evaluate a context-aware access control framework for smartwatch sensors. The framework (Figure 8) consists of two key components: (i) a real-time typing activity detection (rTAD) and (ii) a motion sensor access-controller (MSAC). Preliminary evaluations of the framework lead us to very promising results.

7.1 Typing Activity Recognition

Detecting when a smartwatch user is typing on a keyboard is not as straightforward as detecting contexts such as location or temperature. Running complex machine learning based classification on very limited processors of smartwatches is not a practical solution. Moreover, rTAD must be real-time so that protection measures can be activated proactively. The second bottleneck is the limited battery capacity. Sampling sensors at high frequency and performing complex computations discharges the smartwatch battery rapidly, requiring frequent recharge of the device. For example, continuous sampling of the accelerometer and gyroscope at 50 Hz on our Samsung Gear Live smartwatch completely drains the battery in less than an hour of use, which will severely affect the usability. From these observations it is evident that we have to identify features which are easy to compute and compatible with low sensor sampling rates. However, reducing sampling frequency also means compromising the accuracy of classification. To compensate the reduction in sampling frequency, we design features using a assorted set of motion sensors (sampled at approximately 15 Hz) in order to make a highly perceptive decision. Following are the five feature we incorporate in our proposed rTAD component:

- **Energy:** Activity measured in terms of cumulative linear accelerometer readings. An unworn watch lying on a table has zero energy, while an athlete's watch has high energy. Typing activity typically results in low but non-zero energy. We apply a low pass filter over the linear



(a) N=1 (b) N=2

Figure 9: From bottom to top, (1) the 10 second detection windows where typing was detected are marked in red vertical lines, (2) when N detections occurs within a minute, typing activity is recognized for that 15 minute time segment, and (3) the ground truth collect by prompting the participant.

accelerometer to eliminate high-frequency noise caused by environmental factors.

- **Turnarounds:** Major positive to negative (or vice versa) changes on linear accelerometer readings signify the turnarounds adjoining transitional movements between key presses. Multiple turnarounds in close time proximity can be associated with many activities, such as brushing teeth, eating, playing drums, etc. As a result, we need additional features to distinguish typing from other similar activities.

- **Magnetic Field Change:** Wrists are not rotated significantly when a user types on a QWERTY keyboard, while sitting in front of a stationary desk. Rapid change in north, east and nadir vectors implies non-typing activity.

- **Direction of Gravity:** Gravity generally remains dominant on z-axis of accelerometer while typing on a horizontally placed keyboard. Any major fluctuations or gravity on x-axis or y-axis implies other activities.

- **Step Count:** We assume that the user will be stationary while typing on a computer keyboard. Thus, whenever step count increases, we rule out typing activity.

At the end of every 10 seconds, rTAD conducts a binary classification of weather the user typed in the last 10 seconds or not. All features for the binary classification resets at the starting of the next 10 second window. The cutoff parameters for *Energy* and *Turnarounds* features are calculated using the test data collected in Section 5, whereas cutoff parameters for *Magnetic Field Change* and *Direction of Gravity* features are calculated heuristically. Cutoff parameter for *Step Count* is straightforward, because any increase in the pedometer count indicates walking. The exact cutoff parameters of each feature used in our evaluation of rTAD can be found in Table 2.

Like many other activity recognition problems, there is an inverse relationship between *precision* (number of actual typing instances divided by number of all identified typing instances) and *recall* (number of identified actual typing instances divided by number of actual typing instances), where it is possible to increase one at the cost of reducing the other. A common approach to draw a favorable balance between false positives and false negatives is to 'recognize' an activity only when multiple instances of the activity are detected in close time proximity [23]. However, the use of rTAD is very different than most informative activity detection applications. The purpose of rTAD is to enable countermeasures against keystroke inference attacks as soon as typing activity is identified. In other words, rTAD's goal is to maximize recall, but not to an extent where high false positives start affecting the utility of other non-malicious applications installed on the smartwatch. We evaluate rTAD in two different settings (visually explained in Figure 9):

Table 2: The rTAD's binary classification uses the following parameters. At the end of each 10 second windows, if any of the features are outside these parameter ranges, then non-typing activity is identified, and vice versa.

Feature	Parameters Ranges
Energy	>= 10 and <= 200, after applying low-pass filter
Turnarounds	>= 6
Magnetic Field Change	<= 2 samples with change in north direction
Direction of Gravity	>= 5 samples with fluctuations, or gravity on x-axis or y-axis
Step Count	<= *PreviousStepCount*

- **N=1:** Typing activity is recognized whenever a 10 second window is classified as a typing window. As a result, countermeasures against keystroke inference attacks can be initiated as early as 10 seconds from when the user starts typing.

- **N=2:** Typing activity is recognized when two or more 10 second windows are classified as typing windows within a minute. Countermeasures against keystroke inference attacks can be initiated no sooner than 20 seconds from when the user starts typing.

7.2 Protection

Once rTAD identifies that the user is typing, countermeasures against keystroke inference attacks can be activated automatically in an non-intrusive fashion. And since the protection mechanism is activated only when the user is identified to be typing on a keyboard, the utility of the motion sensors is not affected when user is actively using other applications on the smartwatch (such as playing games that use motion sensors). Such smart protection measures can be undertaken by the MSAC implemented in the operating system itself, or as a trusted middle-ware. For the framework to work, we assume that all third party applications get access

to motion sensor data only via the MSAC and the MSAC has the ability to modify or restrict the flow of motion sensor data. Although this assumption requires change in operating system architecture, it should be a rudimentary task for operating system developers. Also, it should be noted that this assumption does not require changes in existing third party application, as long as the APIs to access motion sensors remain unchanged. Since MSAC requires a change in the operating system architecture, we are unable to implement a working MSAC. However, below we list out some of the strategies that the MSAC can adopt when rTAD reports that the user is typing:

- **Complete Blocking:** This strategy is the safest as it will completely block the side-channel, but it can also harm the utility of other non-malicious applications that may want to perform passive computing with motion data.
- **Reduced Sampling Rate:** When a user types for significant amount of time in a day, complete blocking of the motion sensor data from third party applications can greatly harm the utility of other non-malicious applications. In order to preserve some of the utility, MSAC can provide third party applications access to motion sensors at a reduced sampling rate. Restricting the precision at which applications are allowed to access the sensors reduces the efficiency of side-channel attacks [19, 17].
- **Random Out of Order Blocks:** A smarter MSAC can send out of order blocks of sensor readings to third party applications. Random out of order blocks of sensor data can greatly lower the inference accuracy of side-channel attacks, but may still preserve utility for certain non-malicious application. For example, a daily calorie counter may not be significantly affected by out of order blocks of sensor readings. That is because the calorie count will be accurate as long as all the motions are captured by the application, even if out of order. Size of block and randomization algorithm will play a signification role in determining how much an adversary can recover versus the utility of randomly ordered blocks.

There can be other strategies that the MSAC can adopt as well. We think that it will be best if users are allowed to choose among the MSAC protection strategies, suitable for their personal lifestyles.

7.3 Evaluation

We implement and evaluate our proposed rTAD, because the effectiveness of the entire protection mechanism relies on rTAD. To evaluate rTAD, we use the same smartwatch setup detailed in Section 4.3. Preliminary evaluation involved 4 participants with varied lifestyles wearing the watch for long durations. If the rTAD application does not recognize typing activity, it prompts the participant every 15 minutes to collect ground truth. If the rTAD application does recognize typing activity, it prompts the user immediately for ground truth. In case the user continues to type for long period of time, the rTAD application does not ask the user for ground truth for 15 minutes after the initial detection. This avoids annoyance to the participants and results in equitable ground truth collection. In real usage, the user will not be prompted for ground truth, instead the MSAC will automatically start acting as soon as typing activity is reported by rTAD.

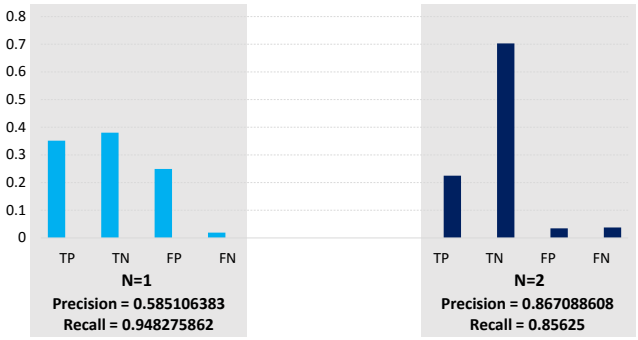


Figure 10: Normalized true positive (TP), true negative (TN), false positive (FP), and false negative (FN), along with precision and recall values.

One problem that we encountered when evaluating rTAD was that in certain cases the *Magnetic Field Change* feature acted unexpectedly, introducing a lot of error in classification. We observed that the unexpected behavior occurred only while the participant typed on a laptop. Further investigation revealed that the magnet inside the laptop's hard drive (which are normally installed directly under the keyboard) was responsible for this unexpected behavior. Since desktop keyboards are generally placed away from the hard drives, the *Magnetic Field Change* feature performed in an expected fashion in that case. For the remainder of the evaluation we do not use the *Magnetic Field Change* feature because it will be hard for the participants to remember and distinguish between laptop and desktop typing. However, as laptops featuring non-magnetic solid state drives are becoming popular, the *Magnetic Field Change* feature may eventually become useful in future.

The combined *true positives (TP)*, *true negatives (TN)*, *false positives (FP)*, and *false negatives (FN)* results from the 4 participants are shown in Figure 10. To better visualize the difference between the two settings, the values in Figure 10 are normalized with respect to the total number of ground truth collected in each setting. As explained with examples in Figure 9, TP signifies that the user was typing and rTAD correctly identified that the user was typing, and if rTAD failed to recognize that the user was typing, it was recorded as FN. Similarly, TN signifies that the user was not typing and rTAD correctly identified that the user was not typing, and if rTAD identified that the user was typing, it was recorded as FP.

In case of N=1, we observe lesser FN and higher TP, but at the cost of higher FP. In case of N=2, we observe lower FP, but at the cost of lower TP and higher FN. In other words, rTAD can gain recall by trading-off precision, and vice versa. Nevertheless, in both settings rTAD achieved high recall values, which asserts it's effectiveness in the protection framework.

7.4 Discussions

- **Left or Right:** Our attack framework, presented earlier in this paper, requires the adversary to have a differently trained framework for targets wearing watch on their right hand. However, the design of rTAD (and thus the whole protection framework) makes it independent of which hand the smartwatch is worn on. As a result, rTAD can start working out of the box, without manual setup.

- **Usability:** Our primary focus while designing the protection framework was usability. We work towards a low processor intensive design, which in turn consumes low battery power. We identify activities similar to typing on keyboard, and try to minimize false positives. The protection mechanism works in an non-intrusive fashion as well, and we envision that the entire setup process in a real-life implementation will be very simple.

8. CONCLUSION

This paper presents a novel keystroke inference attack which utilizes wrist-motion data gathered from a smartwatch as side-channel information. In order to harvest the information masked in wrist movements for inferring keystrokes, we designed and validated a novel learning-based attack framework which is specifically targeted towards recovering text typed by a smartwatch wearing user on an external QWERTY keyboard. By showing the feasibility of the proposed classification and prediction mechanisms, we validate our hypothesis that wearable devices such as smartwatches can leak sensitive personal information if access to sensors (on these devices) is not appropriately regulated. We also present a smart protection framework to automatically regulate sensor access, aimed to improve privacy without degrading utility of the device.

9. ACKNOWLEDGMENTS

Research reported in this publication was partially supported by the Division of Computer and Network Systems (CNS) of the National Science Foundation (NSF) under award number 1523960 and by the Information Institute of the U.S. Air Force Research Lab (AFRL) under the summer faculty fellowship extension grant. The content is solely the responsibility of the authors and does not necessarily represent the official views of the NSF or the AFRL. The authors would also like to thank Dr. Kevin Kwiat and Dr. Charles Kamhoua for their valuable inputs and suggestions.

10. REFERENCES

[1] IEEE Recommended Practices for Speech Quality Measurements. *IEEE Transactions on Audio and Electroacoustics*, 1969.

[2] Experian Marketing Services - Simmons Connect. http://tinyurl.com/experiansmartphones, May 2013. [Online; accessed 8-June-2015].

[3] D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi. The EM Side-channel(s). In *Cryptographic Hardware and Embedded Systems*, 2002.

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

[5] M. Backes, T. Chen, M. Duermuth, H. Lensch, and M. Welk. Tempest in a Teapot: Compromising Reflections Revisited. In *IEEE S&P*, 2009.

[6] M. Backes, M. Dürmuth, S. Gerling, M. Pinkal, and C. Sporleder. Acoustic Side-Channel Attacks on Printers. In *USENIX Security*, 2010.

[7] M. Backes, M. Durmuth, and D. Unruh. Compromising Reflections-or-How to Read LCD Monitors Around the Corner. In *IEEE S&P*, 2008.

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

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

[10] J. Cappos, L. Wang, R. Weiss, Y. Yang, and Y. Zhuang. BlurSense: Dynamic Fine-Grained Access Control for Smartphone Privacy. In *IEEE Sensors Applications Symposium*, 2014.

[11] T. Fiebig, J. Krissler, and R. Hänsch. Security Impact of High Resolution Smartphone Cameras. In *USENIX WOOT*, 2014.

[12] J. Friedman. Tempest: A Signal Problem. *NSA Cryptologic Spectrum*, 1972.

[13] M. G. Kuhn. Optical Time-Domain Eavesdropping Risks of CRT Displays. In *IEEE S&P*, 2002.

[14] M. G. Kuhn and R. J. Anderson. Soft Tempest: Hidden Data Transmission Using Electromagnetic Emanations. In *Information Hiding, Lecture Notes in Computer Science*, 1998.

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

[16] P. Marquardt, A. Verma, H. Carter, and P. Traynor. (sp)iPhone: Decoding Vibrations From Nearby Keyboards Using Mobile Phone Accelerometers. In *ACM CCS*, 2011.

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

[18] L. T. Nguyen, H.-T. Cheng, P. Wu, S. Buthpitiya, and Y. Zhang. PnLUM: System for Prediction of Next Location for Users with Mobility. In *Nokia Mobile Data Challenge Workshop*, 2012.

[19] E. Owusu, J. Han, S. Das, A. Perrig, and J. Zhang. ACCessory: Password Inference using Accelerometers on Smartphones. In *ACM HotMobile*, 2012.

[20] J.-J. Quisquater and D. Samyde. ElectroMagnetic Analysis (EMA): Measures and Countermeasures for Smart Cards. In *Smart Card Programming and Security, Lecture Notes in Computer Science*, 2001.

[21] R. Schlegel, K. Zhang, X.-y. Zhou, M. Intwala, A. Kapadia, and X. Wang. Soundcomber: A Stealthy and Context-Aware Sound Trojan for Smartphones. In *ISOC NDSS*, 2011.

[22] P. Smulders. The Threat of Information Theft by Reception of Electromagnetic Radiation from RS-232 Cables. *Computers & Security*, 9(1), 1990.

[23] E. Thomaz, I. Essa, and G. D. Abowd. A Practical Approach for Recognizing Eating Moments with Wrist-mounted Inertial Sensing. In *ACM UbiComp*, 2015.

[24] W. Van Eck. Electromagnetic Radiation from Video Display Units: An Eavesdropping Risk? *Computers & Security*, 4(4), 1985.

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

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