Inferring Smartphone KeyPress via Smartwatch Inertial Sensing

Sougata Sen[‡], Karan Grover[†], Vigneshwaran Subbaraju[‡], Archan Misra[‡]
[‡]Singapore Management University and [†]IIIT Delhi
[‡]{sougata.sen.2012, vigneshwaran, archanm}@smu.edu.sg, [†]{karan13048}@iiitd.ac.in

***Abstract*—Due to numerous benefits, sensor-rich smartwatches and wrist-worn wearable devices are quickly gaining popularity. The popularity of these devices also raises privacy concerns. In this paper we explore one such privacy concern: the possibility of extracting the location of a user's touch-event on a smartphone, using the inertial sensor data of a smartwatch worn by the user on the same arm. This is a major concern not only because it might be possible for an attacker to extract private and sensitive information from the inputs provided but also because the attack mode utilises a device (smartwatch) that is distinct from the device being attacked (smartphone). Through a user study we find that such attacks are possible. Specifically, we can infer the user's entry pattern on a qwerty keyboard, with an error bound of ±2 neighboring keys, with 73.85% accuracy. As a possible preventive mechanism, we also show that adding a little white noise to inertial sensor data can reduce the inference accuracy by almost 30%, without affecting the accuracy of macro-gesture recognition.**

I. Introduction

Wrist-worn smart devices such as smartwatches (e.g. [5]) and health monitoring devices such as FitBit [3] are becoming increasingly popular. Also the sensors on such devices are being exploited for a variety of innovative applications, such as smoking detection [9] and food journaling [11]. However, such sensor data can also be used to leak personal information. In this paper, we investigate one such possible form of leak - *is it possible to localize the* smartphone *screen touch position using data from the inertial sensors of the* smartwatch *when the user is typing on the* smartphone*? And, if so, are there any low-impact ways to minimise such leakage?* Inferring such screen touch locations on the smartphone can result in leakage of sensitive personal information, such as mobile banking passwords and search keywords in a browser. This is dangerous because the attackers can compromise a smartphone without actually compromising this device.

This problem of attacking one device using another device as a host is, of course, not new. Work such as [12] and [13] have shown that inertial sensor data from the smartwatch can be used to determine key strokes on computer keyboards as well as ATM keyboards. Our work is similar in spirit to such past work, but with one distinctive difference: we investigate a common interaction pattern, where the user wearing the smartwatch is interacting with the handheld smartphone's touch-screen using a single finger. In this situation, the user's forearm exhibits little to no movement, with the touch activity involving just finger displacement. In Figure 1 we can see that even though there is slight wrist movement while typing, however, there is little or no forearm displacement. In contrast, past work has looked at situations where there is a clear displacement of the arm itself during the process of interaction. It is thus unclear if similar inertial sensing-based analysis is useful when a person merely moves a finger around while touching the screen. Moreover, to prevent attacks similar to the ATM pin attack, various precautionary measures can be taken - e.g. disabling the inertial sensors of the smartwatch when a person is at a particular location (e.g. near ATM) or at a particular time. However, such location specific strategies are not useful for preventing leakage that can occur during regular daily smartphone usage.

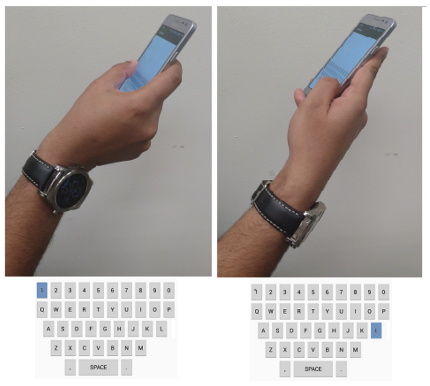


Fig. 1: Wrist position variation for different screen touch events

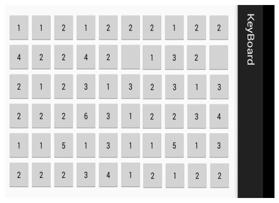
In this work, we investigate the possibility of using inertial sensors to isolate the on-screen location of touch-based interaction by a user with her smartphone. To localise the touch, two challenges have to be addressed: (a) identifying *when* the user touched the smartphone and (b) identifying *where* the user touched. In this work we assume that we know when the touch occurred and focus on identifying the second challenge - where the touch occurred.

Key Contributions: This paper's major contribution includes:

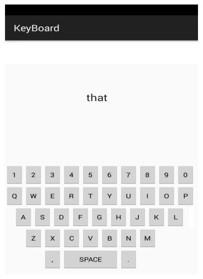
- We show that it is possible to identify the location of a smartphone click using only the inertial sensor data of a smartwatch. More specifically, we extract features from the smartwatch's inertial sensors and identify the features which have the high discriminative power in identifying the location of the on-screen touch events performed by the user.

This research is supported by the National Research Foundation, Prime Minister's Office, Singapore under its IDM Futures Funding Initiative and by Singapore Ministry of Education Academic Research Fund Tier 2 under research grant MOE2014-T2-1063. All findings and recommendations are those of the authors and do not necessarily reflect the views of the granting agency, or Singapore Management University.

978-1-5090-4338-5/17/$31.00 ©2017 IEEE



(a) Grid Mode



(b) Qwerty Mode

Fig. 2: Screenshot of application running on the Smartphone

- Through a small user study, we show that with reasonable accuracy we can identify location of on-screen touch event. For a 4.7″ display divided into 2 partitions, it is possible to identify which partition was touched with an accuracy of 86.32%. For a regular qwerty keyboard, it is possible to identify the key pressed, to within ±2 neighboring keys with an accuracy of 73.85%.

- We also show that adding a small amount of white Gaussian Noise to the inertial sensor data can reduce button inference capability by almost 30%

II. Methodology

In this section, we first describe the data collection application, then the user study details and finally, the data processing pipeline. We start with the application details.

A. Application

For this study, we developed two android applications - a smartwatch application and a smartphone application.

Smartwatch application: The smartwatch application continuously collects the accelerometer and gyroscope data from the smartwatch at the highest sampling frequency. We have used the LG Urbane W150 smartwatch for our studies which has a sampling frequency of 200 Hz for both the sensors. The data from the two sensors is stored locally and is later used to localise the smartphone click.

Smartphone application: There are two data collection modes in the smartphone application - (a) grid mode consisting of 60 buttons which were arranged in 6 rows and 10 columns (Figure 2a) and (b) qwerty mode which consisted of 39 buttons (Figure 2b). All buttons are 6 mm in height with a 1.5 mm gap between buttons. The width of buttons in grid mode is 6 mm, while it is 4 mm in qwerty mode (except spacebar). The phone application collects the ground truth by logging the time key press time and the key release time using Android's *View.onTouchListener()* callback method. For the grid mode, for each button, the button click count information is displayed on the button. Every time a button is clicked, the display is updated to display the incremented button click count on the clicked button. This was useful for our data collection. Participants used the grid mode in landscape orientation and the qwerty mode in portrait orientation.

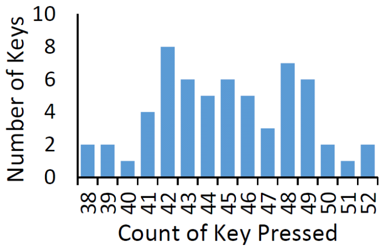


Fig. 3: Distribution of key pressed count

B. Assumptions

Before discussing about the data collection and evaluation, let us discuss the assumptions we have made in this work. With some additional work (discussed in Section V), some of these assumptions might be relaxed.

Firstly, we assume that an application has been installed and is continuously running in the background of the smartwatch to collect the inertial sensor data at the highest sampling frequency. Currently Android does not require any special permission to obtain these sensor readings. Thus, if the user installs the application, this can continuously monitor hand movements. While continuous data collection can cause high energy overhead, we do not consider this issue in our current analysis.

Further, we assume that the participant will be wearing the watch on the dominant hand and will press the keys on the phone with the thumb of the dominant hand, while being seated comfortably. Exceptions to this assumption which we have not tried in our studies are: the user holds the phone with one hand and keys the button with their index finger of the other hand, the person is walking around while pressing the keys, the user uses both hands to input text etc.

In this work we also assume that we obtain the key press and release time from the smartphone and we extract sensor features corresponding to the interval between the key press and release on the phone from the watch. We have discussed various techniques in Section V to overcome this assumption.

C. Data Collection

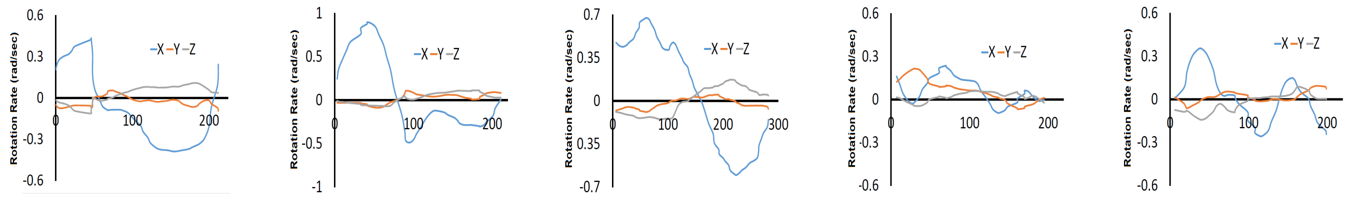
For the study, we collected data from 6 participants. All participant wore the LG Urbane W150 [5] smartwatch on their dominant hand (for all participants, watch face was on the same side as the participant's knuckle) while using a Samsung A3 smartphone [10] (with a 4.7″ display) which had our custom application running. The 6 participants in the study included 4 males and 2 females, who were between 24 to 35 years of age. All participants were right handed and regular smartphone users. Each one of the 6 participants had worn a smartwatch on multiple occasions previously and were comfortable using it. The smartwatch was strapped to the wrist to ensure that the participant was comfortable with the tightness, while at the same time, the watch did not rotate around the wrist.

The data collection was divided into two parts: (a) grid mode data collection and (b) qwerty mode data collection. Data for grid mode was collected from all the participants on one day, while the qwerty mode data was collected on another day.

Grid Mode: For the grid mode, participants were instructed to hold the phone in landscape mode with both hands and press

Seq	Feature	Count	Feature Type	Description
1	Mean	4	S	The mean of the three axis of sensor (3 features) and the mean of their magnitude(1 feature)
2	Variance	4	S	The variance of the three axis of sensor (3 features) and the variance of their magnitude(1 feature)
3	Min	3	S	Minimum sensor reading for each axis
4	Max	3	S	Maximum sensor reading for each axis
5	MCR	3	S	Mean Crossing Rate for each axis
6	Mean_allAxis	1	S	Mean of the summation of sensor values of all axes
7	Mean_axisDistance	3	S	The difference of Mean of (X,Y) axis, (Y,Z) axis and (Z,X) axis
8	Mean_absolute	3	S	mean of absolute of value of each axis
9	Energy	4	S	Energy computed from the FFT of each axis (3 features) and magnitude (1 feature)
10	Duration	1	E	Duration of the button click

TABLE I: The list of features extracted to identify the button pressed. Feature type *S* has been extracted independently for each sensor and feature type *E* has been extracted for an episode.



(a) Top Left (b) Bottom Left (c) Center (d) Top Right (e) Bottom Right

Fig. 4: Change in the three gyroscope axis when user clicks different screen positions

keys in any order using only the thumb of their dominant hand. The participants were instructed to click each key at least 5 times. There was no penalty in case a key was clicked more than 5 times. For this mode, we recorded a total of 2698 key presses for the 60 keys. Since users pressed the keys randomly while ensuring at least 5 entries per key, the total number of clicks each key received varied highly between 38 key presses (for two of the keys) and 52 key presses (again for 2 keys). Figure 3 shows the variation in count of key presses received against the number of buttons receiving the key presses. The average key presses received was ≈45 (7.5 clicks per user).

Qwerty mode: For the qwerty mode, participants were instructed to hold the phone in portrait mode with their dominant hand and type the word that appeared on the screen of the mobile device using the thumb of the same hand. The words were randomly selected from 300 most frequently used words as listed in Word Frequency Data [14]. A total of 30 words were shown on screen. Since the words were randomly picked, the same word could appear multiple times for the same user. The words that were chosen varied between 1 and 10 characters in length and remained on screen till the participant indicated that (s)he had finished typing the word by pressing the spacebar. On spacebar click, the next word appeared on screen. No spell check was done to verify if the participant had typed in the word correctly, as we are currently not trying to recognise individual words. For the 180 words that were typed, we recorded a total of 812 alphabet presses, indicating that the average word length typed was 4.5 characters.

D. Data Processing

We now describe the entire data processing process. The data processing steps are similar to many recent activity recognition or gesture recognition works.

1) Pre-Processing and Framing: At the end of data collection session, the accelerometer and gyroscope data from the smartwatch is extracted and passed through the data processing pipeline. The first step in the data processing pipeline is to remove the gravity component from the data obtained from the accelerometer sensor of the smartwatch. We used techniques mentioned in [7] to eliminate the gravity components. In parallel, key press time from the smartphone is extracted and after synchronising the data from the watch and phone, the smartwatch's accelerometer and gyroscope sensor data corresponding to key press (time obtained from smartphone) is extracted and framed. The entire duration between a key-press and a key-release is considered as one frame. Different instances for the same key press could have varied frame size. For our data we found that on average a key was pressed for 324 millisecond (SD = 168 millisecond) and inter key time is 461 milliseconds, ≈50% than typing on a POS terminal([6]).

2) Feature Extraction: For each frame, features described in Table I is extracted. A feature can either be a sensor specific feature (S) or temporal feature (E) for the frame (column - Feature Type). All the features of type *S* are extracted individually for both the accelerometer as well as the gyroscope sensor. Features of type *S* are computed either for (a) the three axis individually or (b) the three axis and magnitude of the three axis (c) combination of all axis. The description column has further description about each of the features. A total of 57 features (28 for accelerometer, 28 for gyroscope and 1 episode feature) were extracted for our experiments. We computed the information gain for each of the features in our dataset and after ranking them based on their discriminatory power, we found that the features with the highest information gain were: (i) energy of magnitude of gyro, (ii) max of accel's Z axis, (iii) mean of absolute of accel's Y axis, (iv) variance of gyro's X axis and (v) mean of distance between X and Z axis of accel.

3) Key Classification: Once the features for every key press of every user is extracted, we combine the data of all the users together and using a 10-fold cross-validation using the Weka software [4], accuracy of the approach is determined. We evaluated various classifiers and found that the accuracy obtained using a Random Forest classifier is the highest.

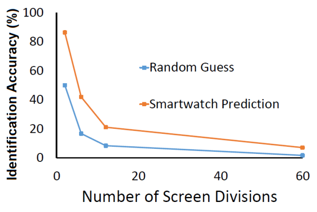


Fig. 5: Prediction accuracy vs. number of screen divisions

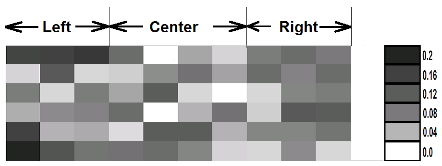


Fig. 6: Heatmap showing which prediction accuracy of different regions

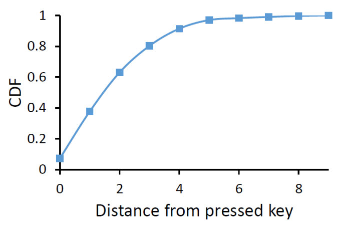
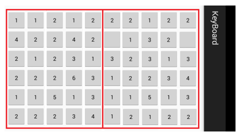
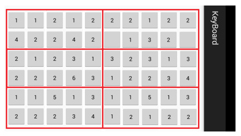


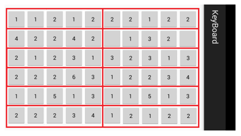
Fig. 7: Error(Distance) of the inferred screen button from actual screen button for Grid mode



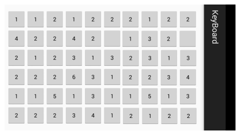
(a) 2 buckets



(b) 6 buckets



(c) 12 buckets



(d) Original (60 buckets)

Fig. 8: Division of screen in grid mode into buckets

We report the results obtained by using the Random Forest classifier for all our studies.

III. Evaluation

The evaluation of our system is divided into two parts - evaluation for the grid mode (landscape orientation) and the evaluation for the qwerty mode (portrait orientation). All the evaluation is done using the data from the 6 users. We first discuss the evaluation for grid mode.

A. Analysis of Landscape Orientation

The 2698 key press data (Section II-C) that was collected from the 6 users was used in the grid mode analysis. Before classifying, we plotted the gyroscope data when a user pressed different positions of the screen. Figure 4 shows the variation of the gyroscope reading when different screen positions are pressed. From the figure we can see that the radians per second variation for X axis is higher when the user is clicking buttons located on the left side of the screen as compared to the right side (user's dominant hand is the right hand).

We used Weka's implementation of Random Forest classifier and performed a 10-fold cross validation to identify which button was pressed. We found that with 7.04% accuracy we could identify which button was pressed (random guess accuracy probability is 1 in 60 button = 1.67%), which indicates a 4.2X improvement over the naïve approach. We had designed our application to have 60 buttons as 60 standard sized buttons filled up the entire screen of the smartphone. However, to leak some personal information, it is not necessary that the entire screen has to be divided into 60 divisions (e.g. identifying which one amongst the 10 digits was clicked can reveal a digit of a PIN or which one amongst 26 alphabets was clicked can leak text messages). Since we could see the difference in sensor data pattern when a key on the left side was pressed versus when a key on the right side was pressed, we analysed the identification accuracy when buttons were clustered together. We divided the 60 buttons into buckets (a bucket consists of one or more buttons and all buckets have the exact number of buttons) of various sizes (as shown in Figure 8) and computed the prediction accuracy. Figure 5 shows the accuracy of predicting the correct bucket that was clicked. We found that when we divide the screen into 2 buckets, where each bucket has 30 buttons (refer to Figure 8a), the overall system accuracy is 86.32% (1.72X improvement over naïve). On computing the prediction accuracy for different bucket sizes we see that when the number of buckets to be classified are larger (more number of classes), the improvement of our classification technique over the random guess approach is higher.

For a screen with 60 divisions, we found that the average classification accuracy was 7.04%. Does this mean that every division in the screen has a 7.04% accuracy? To answer this question, we computed the classification accuracy for each button. Figure 6 shows the heatmap of prediction accuracy of each button. In the figure, divisions on the top left corner represent the buttons on the top left corner of the physical device which our participants used. From the figure we see that identification of keys towards the left or right corners is better than that in the center. In fact, the prediction accuracy for some keys on the left side is above 19%. Amongst the keys in the center, the keys towards the bottom get higher prediction accuracy as compared to the keys on the upper part of the center screen.

In over 92% cases, we fail to identify the exact button (6mm x 6mm). Given this, we next investigated the distance between the predicted and the true button pressed. The distance was computed based on how many buttons separated the actual button from the predicted button. Buttons that were adjacent got a distance of 1, while keys that were 2 hops away were given a distance of 2 and so on. For example, based on this calculation, the distance between (5,3) and (7,6) is 3.$\{(5,3) \xrightarrow{\textcircled{1}} (6,4) \xrightarrow{\textcircled{2}} (7,5) \xrightarrow{\textcircled{3}} (7,6)\}$. Figure 7 shows the CDF of the distance between predicted button and actual button. From the CDF we can see that for a screen with 60 buttons, in 63.08% cases, the error distance was 2 buttons or less indicating that in most cases we could identify approximately which part of the screen was pressed.

Finally, we wanted to understand the impact of personalised models. It is well known that a personalized classification model usually perform better than a general model trained on multiple user's data. Accuracy results indicated above was

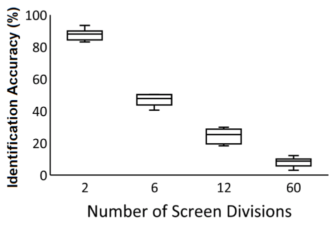


Fig. 9: Variation in accuracy for a personalized model

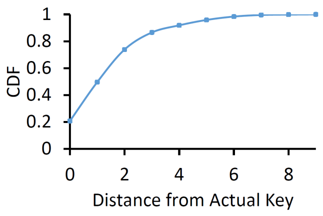


Fig. 10: CDF of distance from actual key press in a qwerty keyboard

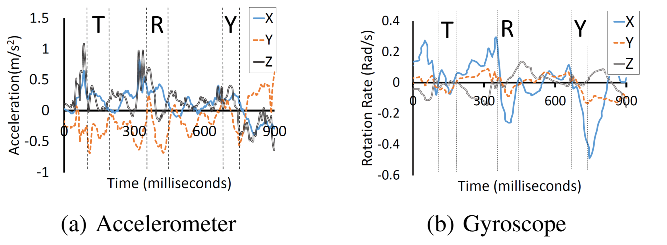


Fig. 11: Variation of accelerometer and gyroscope sensor data for one participant typing *TRY*

obtained by cross validating the results of the 6 users. We next computed the classification accuracy of users individually and found that for 60 divisions, the median of individual user accuracy was 8.72%. Figure 9 shows the box plot for prediction accuracy on a per user data for different screen divisions. From the plot we observe that for most screen divisions, the per-user accuracy obtained for most users is better than the accuracy obtained when a common model is used.

B. Analysis of Portrait Orientation

For the portrait orientation, we used the data extracted from the entries of 180 words obtained from 6 users. We plotted the accelerometer and gyroscope values when one of the users typed the word *TRY*. Figure 11a shows the variation of the accelerometer readings when a user typed, while Figure 11b shows a variation of the gyroscope readings. From the plots, we can see that similar to the grid mode, there was visible variation when user typed in the qwerty mode. For a more detailed analysis, 812 alphabets were keyed in by the users while keying in the 180 words. These alphabets were used for our analysis and we performed a 10-fold cross-validation on features extract for these alphabets. Classification was performed with the same features as mentioned in Table I, using a Random Forest classifier. On classifying, we achieved an accuracy of 20.67% when the class labels were the 26 English alphabets. This indicates a 5.37X improvement over random key prediction ($(1/26)*100 = 3.84\%$).

Similar to the distance estimation in grid mode, we computed the error in distance estimation to assess our prediction accuracy. Since the keys in the three rows of the qwerty keyboard are not aligned, we shifted the position of the alphabet *A* so that it was considered to be in the same column as *Q* and we aligned the alphabet *Z* so that it was considered to be in the same column as *W*. From the results we found that in 49.63% cases we had a ± 1 key estimation error, which increased to 73.85% for a ± 2 estimation error. The CDF of error in key distance prediction for the qwerty keyboard is plotted in Figure 10. From the plot we can see that the error in 49.63% cases is within 1 button indicating that if alphabet pairs are carefully pruned, one might identify words being typed based on alphabet press sequence and using a threshold edit distance to identify wrongly keyed alphabets.

IV. Effect of Adding Noise

Since inference of click event location might lead to various side channel attacks, we wanted to understand if this threat could be mitigated by adding noise to the data. We added two types of noise to the data collected for the grid mode: (a) Gaussian white noise was added to the entire time series of the inertial sensor data which was collected from the participants and (b) after sensor readings from the inertial sensors were extracted for a particular key press, for k key presses, we added n random sensor readings (extracted from the sensor data of the same user) to the sensor readings for the key press.

We extracted features for the new sensor readings and performed a 10-fold cross validation on the entire dataset of 2698 key presses. For noise type (a), white Gaussian noise was added to the original inertial sensor readings at an signal to noise ratio of 20dB and 30dB. We found that for 20dB, the accuracy of the system dropped from 7.04% to 4.98%. In case of 30dB, the accuracy was at 5.74%, indicating that adding little noise (30dB) causes an 18% drop in accuracy and moderate noise (20dB) caused almost 30% drop in accuracy. For noise type (b), for 20% of key presses ($k = 540$) we added 1 random reading ($n = 1$) and found that the accuracy dropped to 5.8% and when we increased the value of k to 2158 (80% key presses) and added 1 random readings, the inference accuracy was at 5.04%. This indicated that either adding Gaussian noise to the entire sensor reading from the smartwatch or adding random sensor values to the data can help in reducing the threat of a side channel attack. However, there might be other useful applications running on the smartwatch (e.g. [9], [11]) which utilize the same inertial sensors and should not be affected by the noise. Since these applications look at a larger data window as compared to key press inference, the noise that were introduced should have little effect on their performance.

V. Discussion and Future Work

In this work we show that it is possible to identify the location on a smartphone's screen that was clicked using inertial sensor data from a smartwatch. However, this is just an initial work and to extend this as a fully working system, various directions have to be explored. In this section we discuss some of those directions.

Distinguishing Key Presses: In this current work we extract key press times from the ground truth file and extract the sensor data corresponding to that window. However, to make the system realistic, we will have to automatically determine the end of one key press and the starting of the next key press. Work such as [2] has used microphone to distinguish between alphabets when a person writes on a whiteboard, while [13] uses the z axis movement to make this determination. We could evaluate techniques similar to these ideas for our current work. Alternately, an application running on the phone might be able to use the phone's vibration when a user types to determine the key press and release time. Since inertial sensing does not require additional permissions, this application can collect the data while running in the background.

Two hand Typing: Currently we have assumed that the user types with one hand using the thumb. This might not be a valid assumption in certain cases, especially when the user is holding the phone in landscape orientation and types with both hands). In such cases we can use the inter word gap to determine if a word has been pressed. For example, in case the user is wearing the watch in the right hand and is typing the word *PLAY*, where the user uses the right hand to type *P*, *L* and *Y* and the left hand to type *A*. Since the inter alphabet gap is predictable based on historical data (user takes average of *486* milliseconds to release *L* and press *Y*), if the gap between alphabets pressed is more than the threshold gap, the multiple of the threshold gap between the two alphabets can indicate the number of alphabets keyed in with the non-watch hand.

Useful Application: Being able to identify screen click locations might cause privacy concerns. However, that should not prevent us from using it for useful scenarios. Various augmented reality (AR) applications as well as virtual reality (VR) applications can utilise this click localisation technique as a new virtual input modality. An example of the usability in VR applications might be if a virtual keyboard appears on the VR display and the user has to type in her inputs. In such a scenario, if the user is wearing a smartwatch and holding onto a smartphone, then based on the wrist movement (detected through smartwatch), the VR might be able to infer the button towards which the hand is moving and on identifying click, it can identify the user intention. Similarly, in case of AR applications, inferring what a user types using data from the smartwatch might be an alternate technique to camera based inference.

VI. Related Work

Using motion sensors on devices such as smartphones and smartwatches to interpret gestural movements of the hand have been studied in the literature for various purposes. The *PhonePoint Pen* prototype [1] used the on board accelerometer in smartphones to recognize human writing in the air. The advent of smartwatches made the recognition of such hand gestures much more accurate and easier. In [15], the accelerometer and gyroscope sensors of a wrist worn device was used to recognize finger gestures such as 'Index finger click', 'zoom in' and 'zoom out' with a high level of accuracy. A similar approach was also used to recognize the alphabet traced by the user's finger on a surface with an accuracy of 95%.

Wang et al. in [12] showed that the motion sensors on wrist worn devices can be used to reproduce the trajectories of the user's hand movements thereby presenting and opportunity to reveal the secret key entries. However, the study focused on surfaces such as key pads on ATMs and keyboards. Similarly, [2] showed that motion sensor data collected on a smartwatch may be used to infer text written on a white board. [13] and [6] again used a similar approaches to detect keys pressed on laptop keyboards. But these studies did not consider small surfaces such as smartphone keypads, which are held in the same hand as the smartwatch and where the range of motion is typically limited to just finger movements.

The work in [8] may be the one that is most similar to the approach used in this paper. In [8], the accelerometer in the smartphone was used to identify the key and the on-screen location that was pressed on the smartphone's keypad. This study showed that the approach may be used to expose user's passwords and hence represents a security hazard. Approaches to mitigate this security risk were also discussed in [8]. However, our study addresses a different problem, where the sensors of the smartwatch are used to infer the keypress events on the smartphone. The consequent security risk in even more difficult to mitigate since the malware is not present on the device that is being compromised.

VII. Conclusion

In this paper we show that it is possible to localize the screen click position on a smartphone using the inertial sensor data from a smartwatch with a reasonable accuracy. Through a small user study we have shown that we could achieve more than 4.2X improvement in predicting the location of a click over the naïve prediction approach. In the case of a qwerty keyboard, our alphabet prediction accuracy was above 20% for a 26 class classification; however the accuracy jumps to 73% when we allow a tolerance of ± 2 keys. However, the addition of a modest amount of noise proves effective against such eavesdropping attacks.

References

[1] Agrawal, S., Constandache, I., Gaonkar, S., Roy Choudhury, R., Caves, K., and DeRuyter, F. Using mobile phones to write in air. *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services*, MobiSys '11.

[2] Ardser, L., Bissig, P., Brandes, P., and Wattenhofer, R. Recognizing text using motion data from a smartwatch. *Pervasive Computing and Communication Workshops (PerCom Workshops)*, 2016.

[3] Fitbit. http://www.fitbit.com/, November 2016.

[4] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 2009.

[5] Lg urbane w150. http://www.lg.com/us/smart-watches/lg-W150-lg-watch-urbane, November 2016.

[6] Liu, X., Zhou, Z., Diao, W., Li, Z., and Zhang, K. When good becomes evil: Keystroke inference with smartwatch. *Proceedings of 22nd Conference on Computer and Communications Security*, CCS '15.

[7] Mizell, D. Using gravity to estimate accelerometer orientation. *Proc. 7th IEEE Int. Symposium on Wearable Computers ISWC*, 2003.

[8] Owusu, E., Han, J., Das, S., Perrig, A., and Zhang, J. Accessory: Password inference using accelerometers on smartphones. *12th Workshop on Mobile Computing Systems & Applications*, HotMobile '12.

[9] Parate, A., Chiu, M.-C., Chadowitz, C., Ganesan, D., and Kalogerakis, E. Risq: Recognizing smoking gestures with inertial sensors on a wristband. *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '14.

[10] Samsung a3. http://www.samsung.com/sg/smartphones/galaxy-a3-2016-a310/SM-A310FZDDXSP/, November 2016.

[11] Sen, S., Subbaraju, V., Misra, A., Balan, R. K., and Lee, Y. The case for smartwatch-based diet monitoring. *Pervasive Computing and Communication Workshops (PerCom Workshops), 2015.*

[12] Wang, C., Guo, X., Wang, Y., Chen, Y., and Liu, B. Friend or foe?: Your wearable devices reveal your personal pin. *Proceedings of 11th Asia Conference on Computer and Communications Security*, ASIA CCS'16.

[13] Wang, H., Lai, T. T.-T., and Roy Choudhury, R. Mole: Motion leaks through smartwatch sensors. *Proceedings of 21st Annual International Conference on Mobile Computing and Networking*, MobiCom '15.

[14] Word frequency data. http://www.wordfrequency.info/, November 2016.

[15] Xu, C., Pathak, P. H., and Mohapatra, P. 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*, HotMobile '15.