**Journal of Ambient Intelligence and Humanized Computing manuscript No.**
(will be inserted by the editor)

# AlphaLogger: Detecting Motion-based Side-Channel Attack Using Smartphone Keystrokes

**Abdul Rehman Javed · Mirza Omer Beg · Muhammad Asim · Thar Baker · Ali Hilal Al-Bayatti**

Received: date / Accepted: date

**Abstract** Due to the advancement in technologies and excessive usability of smartphones in various domains (e.g., mobile banking), smartphones became more prone to malicious attacks.Typing on the soft keyboard of a smartphone produces different vibrations, which can be abused to recognize the keys being pressed, hence, facilitating side-channel attacks. In this work, we develop and evaluate *AlphaLogger*- an Android-based application that infers the alphabet keys being typed on a soft keyboard. *AlphaLogger* runs in the background and collects data at a frequency of 10Hz/sec from the smartphone hardware sensors (*accelerometer, gyroscope* and *magnetometer*) to accurately infer the keystrokes being typed on the soft keyboard of all other applications running in the foreground. We show a performance analysis of the different combinations of sensors. A thorough evaluation demonstrates that keystrokes can be inferred with an accuracy of 90.2% using accelerometer, gyroscope, and magnetometer.

**Keywords** Smartphone Security · Keystroke Inference · Side-Channel Attacks · Machine Learning · Motion Sensor

Abdul Rehman Javed
National University of Computer and Emerging Sciences, Islamabad, 44000
Tel.: +92-307-4504643
E-mail: abdurrahman.j74@gmail.com

Dr. Mirza Omer Beg
E-mail: omer.beg@nu.edu.pk
National University of Computer and Emerging Sciences, Islamabad, 44000

Dr. Muhammad Asim
E-mail: muhammad.asim@nu.edu.pk
National University of Computer and Emerging Sciences, Islamabad, 44000

Dr. Thar Baker Shamsa
E-mail: T.M.Shamsa@ljmu.ac.uk
Liverpool John Moores University (LJMU), UK

Ali Hilal Al-Bayatti
E-mail: m_usman296@hotmail.com
De Montfort University, Leicester, UK

## 1 Introduction

Recent expansion in mobile technology has brought about the enhancement of diverse and more powerful sensors-based smartphones that are used for daily activities, such as communication, social interaction, business, and financial transactions (Cook 2010; Deb et al. 2020). Smartphones are embedded with numerous sensors, including Global Positioning System (GPS), audio sensors, motion sensors, light sensors, position sensors, and temperature sensors (Voicu et al. 2019). The availability of readings from these sensors creates exciting new applications, such as remote health monitoring (Hussain et al. 2016), and new concerns for security experts. Since smartphones contain potentially sensitive personal information about the user's activities, attackers are also investing huge amounts of time and effort to create malicious applications to acquire the victim's data (Lanette and Mazmanian 2018). According to a study (Cai and Chen 2011), the W3C DeviceOrientation Event Specification allows applications to access smartphone hardware sensors using Javascript that is supported by both Android 3.0 and iOS 4.2. The Android platform allows applications to read from a vast variety of smartphone sensors, while iOS has a much stricter policy that allows few third-party applications to read hardware sensors. In terms of human-computer interaction, the keyboard is the key input device used to input data and commands in smartphones. Keyboards are commonly used to enter personal data such as PINs, passwords, and credit card information other than the usual text (such as text messages, emails, etc.) (Kucukyilmaz et al. 2008; Tang et al. 2014). However, this utility can make the smartphones vulnerable to attacks such as keylogger side-channel attacks (Hussain et al. 2016). In these attacks, hackers can make users install malicious apps, integrated with keyloggers, to record the keys a user types on a soft keyboard. The leakage of such information can cause serious damage and may lead to loss of confidential information, or financial loss.

Accelerometer and gyroscope readings can be easily accessed using Javascript (Cai and Chen 2011) which is supported by both Android and iOS systems. In this paper, we introduce *AlphaLogger*, which has the capability to infer the soft keys of smartphones using motion sensors. Once the user installs *AlphaLogger* and grants it the motion sensor privilege, it starts the process of sensing motions and inferring keystrokes. We use accelerometer, gyroscope, and magnetometer readings to detect user keystrokes to emulate side-channel attacks. We observe that by using these sensor readings a learning model can easily be trained to accurately determine soft keyboard inputs on an Android device. We develop an Android-based application called *AlphaLogger* that is installed on a user's smartphone. A wide variety of smartphones are available in the market, we make this application run-able for Android-based smartphones.

Although there exist several studies (Ping et al. 2015; Cai and Chen 2011), which look at keystroke inference using smartphone sensors, they lack in providing promising results in regards to achieving higher accuracy when classifying keystrokes. Moreover, they do not consider the keystroke inference across applications (Ping et al. 2015). The results gained from the experimentation of the proposed approach demonstrate a superior performance with an accuracy of 90.2% as compared to the state-of-the-art.

To this end, the main contributions of this paper are as follow:

1. We develop *AlphaLogger*, a systematic and functional application that demonstrates the feasibility of inferring soft keyboard inputs accurately from smartphone sensor readings.
2. We develop a dataset of ten users using different Android smartphones, which provides the diversity to the inferring model to infer the alphabets typed by the smartphone user.
3. We evaluate *AlphaLogger* using state-of-the-art machine learning techniques, including Ensemble Adaboost, Ensemble Voting, Decision Tree (J48), Sequential Minimal Optimization (SMO) and Multilayer Perceptron (MLP). Our results show that the *AlphaLogger* achieves better accuracy than the state-of-the-art.

The paper is organized into five sections. Section 2 briefly covers the technical background and recent advancements in side-channel attacks. Section 3 presents the overview of the proposed *Alphaloger* application. The experimental set up and results are articulated in Section 4. Finally, Section 5 concludes the paper and identifies directions for future work.

## 2 Related Work

In research literature, side-channel attacks have been studied ranging from traditional desktops (Vuagnoux and Pasini 2009; Zhuang et al. 2009; Foo Kune and Kim 2010) to smartphones (Cai and Chen 2011; Xu et al. 2009; Owusu et al. 2012).

Cai et al. (2009) present three studies one after another to examine the motion-based side-channel attacks. In their study, the authors inspect the security ramifications of implicit sensors in smartphones. They talk about a general structure of protection against sensor-sniffing attacks. The work demonstrates the more common sensors such as GPS, cameras, and mouthpieces. The same authors present their first study in (Cai and Chen 2011) to discuss motion-based side-channel attacks. The authors present an Android application called TouchLogger to show the vulnerability of a side-channel attack. TouchLogger utilized machine learning algorithms to infer keystrokes using the gyroscope sensor's reading. The work was assessed on an HTC Evo 4G smartphone in the landscape mode utilizing a numeric keypad. TouchLogger accurately inferred over 70% of keystrokes.

Owusu et al. (2012) present an application to detect keystrokes that divide the smartphone screen into multiple zones and keystroke were inferred using the tapped zone. They use a single accelerometer to infer the readings of a motion sensor. They reported that dividing the smartphone screen into eight zones produces the prediction accuracy of 24%. Xu et al. (2012) perform the online training and classification to track the motion readings. They focused on the numeric keypad to extract pins and passwords. They focus on two types of attacks: the number is written while pressing during a phone call and the other was the lock screen password. They use an accelerometer and gyroscope to collect data for this process. They show that the best PIN inference can be done when a PIN consists of four digits. In Aviv et al. (2012), the authors focus on two modes of passwords: Pin-lock and swipe password. This study was carried out on 24 individuals for each mode. A total of 12 individuals were considered to type the PIN and provide the swiped password using four smartphones. They use a single accelerometer sensor for data

collection. Their study yield an accuracy of 43% when an individual was sitting while typing. They report an accuracy of 20% for the PINs and 40% for the swipe patterns within 5 attempts in a random environment. Ping et al. (2015) present a study for a longer information derivation, for example, chat and email content. They achieve an accuracy of 36% using ensemble learning of four algorithms: Simple Logistic, Random Forest, SMO, and k-Nearest Neighbor. Song et al. (2018) present an algorithm to extract frequency domain features from a motion sensor raw readings to infer the keystroke on a smartphone. Their study reports that PINs and passwords can be inferred efficiently in complex scenarios even when the frequency rate is lower than 80Hz. They report the overall accuracy of 74.6%. A similar study presented by Tang et al. (2018) for inferring user-independent keystrokes to unlock a smartphone. They use a probabilistic model to classify the keystrokes and used the angle of keystroke movements to show the trends in a dynamic environment. They report an accuracy of 70% and 85% in 10 attempts for both user-dependent and user-independent scenarios.

Shumailov et al. (2019) present a study to infer the typed keys on the soft keyboard of the smartphone. They used acoustic signals to predict the typed soft key. They show that the microphone is capable to hear low sound waves which can be translated to recover the alphabet. In results, they show that their approach recovers 61% of the alphabets. Another study by Wang et al. (2019) shows that how a password can be inference using an eye pattern. They use the smartphone's front camera to record the pattern of the user's eyes. They use these patterns to infer the typed password.

Authors in different studies discussed various verification and theft anticipation schemes to counter security attacks. For example, Ali et al. (2018) propose a three-factor based remote client confirmation convention for wireless medical sensor networks to manage off-line password guessing attack, client impersonation assault, session-key temporary information attack and the disclosure of secret parameters. Xu et al. (2019) present an elliptic bend cryptography (ECC) based three-factor authentication scheme for enhancing security in multi-server environments. Kuppusamy (2019) propose two transformations of the password termed as "PassContext" and "PassActions", to counter vulnerabilities in plain-content secret phrases by utilizing the complexities of human-computer association. Ruan et al. (2019) proposed a security planning model for the Three-party password-based authenticated key exchange (3PAKE) protocol conventions that is vulnerable to leakage attacks.

In summary, the accuracy reported in the past papers is quite low and is limited to recognize keystrokes within the applications. Some previous work focuses more on dividing the screen into multiple zones, which fails when the size of a screen varies from device to device. Our proposed framework is highly accurate and supports cross-application keystroke recognition using fused sensors technique.

## 3 AlphaLogger

This section presents the steps of our proposed approach; *keystroke data collection and pre-processing, feature generation and transformation, tap event detection and then machine learning algorithm for inferring keystrokes*, as illustrated in Figure 1.

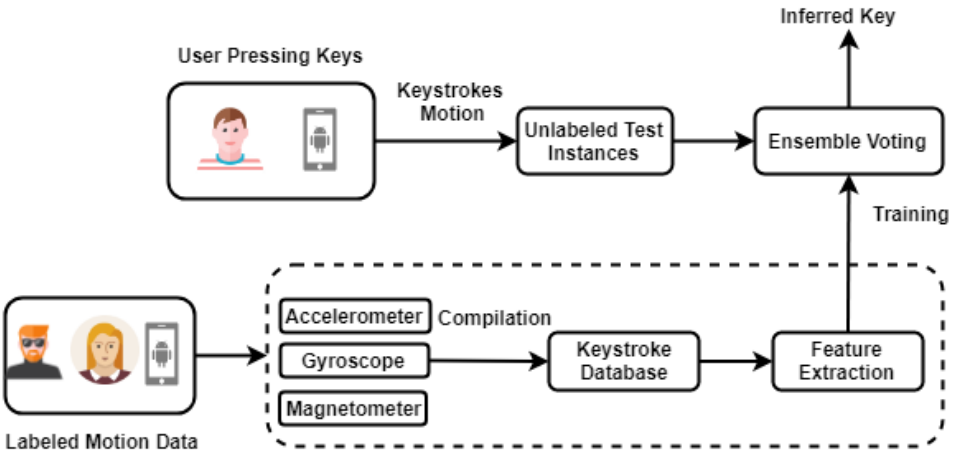



Fig. 1: Block Diagram of the *AlphaLogger*

### 3.1 Keystroke Data Collection and Pre-processing

The data collection task is controlled by an application we created that is installed on an Android-based smartphone. We collect data from 5 different smartphones and ten participants: *Oppo F3, Oppo F1, Samsung J7, Samsung Grand Prime* and *Huawie Honor*. The typing process is performed by the participants during the standing and sitting positions. These are the most commonly used postures with minimum noise generally caused by the body movement. Throughout the experimentation phase, the portrait mode is used, and the participant is asked to hold the smartphone with both hands and type with thumbs. The data collection is performed with a focus on what data to collect and how frequently it needs to be collected. The data is collected at a fixed sample frequency of 30 samples per second as recommended by (Kwapisz et al. 2011; Krause et al. 2005). The collection duration was approximately 2-3 minutes to catch all signals.

The dataset consists of sensor events generated while typing alphabets on a smartphone keyboard. Each smartphone contains various hardware sensors. Some smartphones of our participants were not equipped with the magnetometer sensor and some smartphones were used without the gyroscope sensor. To address this, multiple models are trained: one on the dataset containing raw acceleration readings, and the others are based on readings in combination with other sensors (magnetometer and gyroscope) (see Table 1 for more details). Each alphabet is typed continuously for approximately 2-3 minutes, and all the readings are recorded in a comma-separated file (CSV). To ensure that the recorded readings are well-structured, we assign a timestamp to each reading. In this way, the dataset consists of 26 alphabet files.

### 3.2 Feature Extraction

We transform raw data to a sensor event window. We select a window of 500 samples from each file of each participant. The selected window is diverse enough to capture all the required readings to apply classification methods. We assign labels manually according to the alphabets being pressed as a ground truth. This enables us to correctly map and record the sensor measurements along with the corresponding alphabets being pressed. Later, a feature matrix based on 130,000 raw sensor data readings is generated where each reading contains 3-axis of all the three sensors.

### 3.3 Machine Learning Model

We use the extracted features to build a machine learning model. We decided to use the Weka (Hall et al. 2009) toolkit for training our model to infer the keystrokes. Inference refers to the classification of the alphabets. We use machine learning techniques: Decision Tree (J48), Sequential Minimal Optimization (SMO), and Multilayer Perceptron (MLP) and meta-classifiers Ensemble Adaboost and Ensemble Voting for supervised classification. The choice of machine learning techniques depends on the size of data. Some techniques require a large amount of data, and some can work well with significantly less. Some techniques are designed to work with categorical data and some to work only with numeric data. We use these algorithms to justify the keystroke inference accuracy and results as these algorithms belong to different categories and have a different structure. All these algorithms provide a similar inference accuracy that justifies our approach. We describe the working of algorithms in the following subsections.

- **Decision Tree (J48)**: This classifier builds the decision tree based on their information gain and entropy (Hall et al. 2009). Data is split on each node into subsets based on the highest information gain. A stopping condition is made to stop splitting the tree to control the depth of a tree when the required results are achieved. There are two measures in decision tree: Entropy and Information Gain. For best performance, entropy should be low and information gain should be high. Entropy is calculated as:

$$Entropy = H(T) = I_E(k_1, k_2, ..., k_J) = -\sum_{i=1}^{J} k_i \log_2 k_i \tag{1}$$

  where $Ki$ is the probability of class $i$ in the database. $H$(feature) is the entropy that measures the degree of "impurity". In the dataset, noisy sensor readings that typically occur before and after keystroke, are referred to as impurity. Impurity produces misleading information in the keystroke inference process. The amount of impurity can be estimated by examining $H$. The value of $H$ closer to 0 means the lesser the impurity in the dataset. A good feature provides high information gain and less entropy.

$$IG(F_i) = H(C) - H(C|Fi) \tag{2}$$

  To measure the information gain of a feature $F_i$, $IG$ is calculated as per equation (2) where C represents different classification classes, and $F_i$ represents different features in the dataset.

– **Sequential Minimal Optimization (SMO):** The Support Vector Machine (SVM) algorithm uses the quadratic programming (QP) as an inner loop due to which SMO breaks QP into a series of small QP problems (Platt 1998). It solves the smallest possible optimization sub-problem analytically at each step. The benefit of using SMO is that QP optimization can be avoided entirely which makes it fast to solve sub-problems. The optimization function is given by:

$$\begin{aligned} &\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j K(x_i, x_j) \alpha_i \alpha_j, \\ &\quad subject\ to: \\ &\quad 0 \leq \alpha_i \leq C, \quad for\ i = 1, 2, ..., n, \\ &\qquad\qquad \sum_{i=1}^{n} y_i \alpha_1 = 0 \end{aligned} \tag{3}$$

where $C$ is a SVM hyper-parameter and $K(x-i, x-j)$ is the kernel function, both supplied by the user; and the variables $\alpha_i$ are Lagrange multipliers. In SMO two multipliers are solved first. In the case of SVM, the constraints are changed to the following:

$$\begin{aligned} 0 \leq \alpha_i, \alpha_i \leq C \\ y_1 \alpha_1 + y_2 \alpha_2 = K \end{aligned} \tag{4}$$

– **Multilayer Perceptron (MLP):** MLP is a network of perceptron (Hall et al. 2009). MLP trained with backpropagation algorithm is used for data mining. It uses backpropagation to learn a multilayer perceptron to classify instances. It consists of an activation function, an input layer, hidden layers and output layer which are connected. Each connection consists of a weight. Each node measures the weighted sum of all the inputs and uses threshold model data. Below are the two activation functions: tanh and sigmoid.

$$y(v_i) = \tanh(v_i) \quad \text{and} \quad y(v_i) = (1 + e^{-v_i})^{-1} \tag{5}$$

The node weights are adjusted based on the corrections that minimize the error in the entire output, given by

$$\mathcal{E}(n) = \frac{1}{2} \sum_{j} e_j^2(n) \tag{6}$$

where weight can be updated using gradient decent. The gradient decent function described using the equation:

$$\Delta w_{ji}(n) = -\eta \frac{\partial \mathcal{E}(n)}{\partial v_j(n)} y_i(n) \tag{7}$$

where $y_i$ is the output of the previous perception and $\eta$ is the learning rate.

– **Ensemble Voting**: We use voting ensemble method (Dietterich 2000) for cognitive health classification. It can be used for both supervised and unsupervised learning. It consists of a base classifier and sub-classifiers. Each model makes a prediction and these predictions are combined by several methods like majority voting, average, mean, mode, etc. We picked J48 (Hall et al. 2009), K-nearest

neighbor (kNN) (Altman 1992), SMO (Platt 1998) and MLP (Hall et al. 2009) for voting predictions and used majority voting rule to evaluate results. We got the best results using information gain feature selection, SMOTE data balancing and Adaboost classifier.

– **Ensemble AdaBoost:** AdaBoost is an ensemble method (Dietterich 2000; Graczyk et al. 2010) that works on nominal class attributes for classification. It boosts the performance of weak learner algorithms. It tries to overcome the prediction error made by the classification model. AdaBoost greedily minimizes exponential loss which is defined as:

$$F_T(x) = \sum_{t=1}^{T} f_t(x) \tag{8}$$

where each $f_t$ represents the weak learner that takes an object x. The error is given by:

$$E_t = \sum_{i} E[F_{t-1}(x_i) + \alpha_t h(x_i)] \tag{9}$$

where $h(x_i)$ is the hypothesis that is created by each weak leaner. $\alpha_t$ is the coefficient assigned to each weak learner such that the sum of training error is minimized.

## 4 Evaluation

The experimentation first requires the labeled raw data from all the sensors discussed in Section 3.1, and then transform this data into examples. We collect labeled data from smartphone sensors and then preprocess this data to remove the noise present at the start and end of the dataset. The noise is created at the time when the sensors start acquiring readings exactly when the application comes to play in the foreground. The experimentation is performed using the WEKA data mining tool (Hall et al. 2009) using Ten-fold cross-validation.

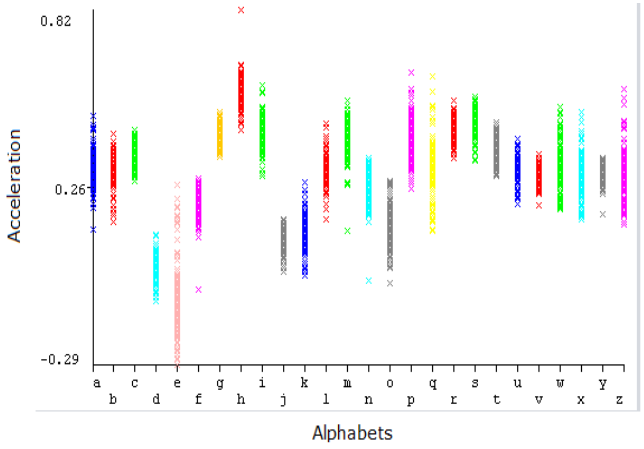
We show the results when an accelerometer is used alone, when it is used in combination with the magnetometer, when it is used in combination with magnetometer and gyroscope, when it is used in combination with the gyroscope and when only the accelerometer readings are used, to show the effectiveness of this additional step. We collect keystrokes data on an alphabet-only soft keyboard. The dataset consists of multiple sessions containing 100 to 200 consecutive keystrokes which are about 2-4 minutes continuously. The datasets contain all the 26 keys of the smartphone keyboard. We use these keystrokes dataset to train and evaluate *AlphaLogger*. To explore a distinct typing environment, the participants were asked to type in two environments, first, writing text messages in the default messaging application and other is posting emails via Gmail. Both types of messages are sensitive and private.

### 4.1 Simple Pattern Analysis

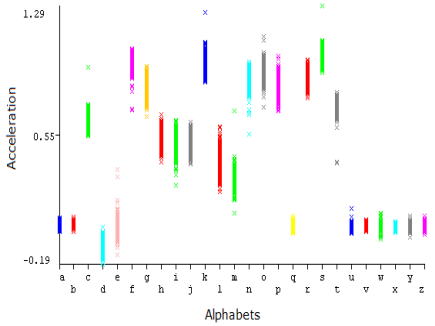
Pattern analysis is an efficient way to understand the behavior of a particular feature. A feature is good if it is consistent among signals produced by the same

keystroke while being distinctive between signals caused by different keystrokes in the case of each hardware sensor: accelerometer, gyroscope, and magnetometer. Figure 2, 3 and 4 describes the analysis of each signal produced by each sensor. We analyze different cases, their strengths, weaknesses and why there is a low performance of our model when it uses with the only accelerometer and when it is used in combination with other sensors.

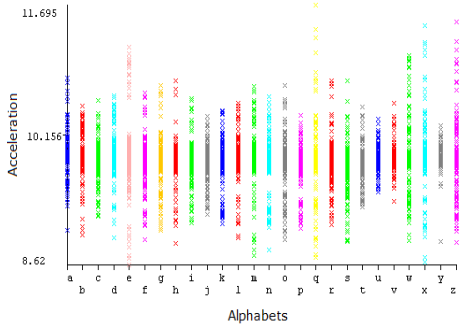
Figures 2a, 2b, 2c respectively show the pattern of three-axis of accelerometer $A_x, A_y, A_z$. It does not show a unique pattern against each character which is the reason for the lower performance of this model and $A_z$ does not contribute in improving the performance of the classifier as it shows the z-axis of an accelerometer which is usable in the case when there is rotation while pressing keys. Furthermore, when we use the accelerometer in combination with magnetometer it starts showing a unique pattern for each character, which results in an improvement in the performance of our model.




(a) Unique Accelerometer $A_x$ Dimension Pattern



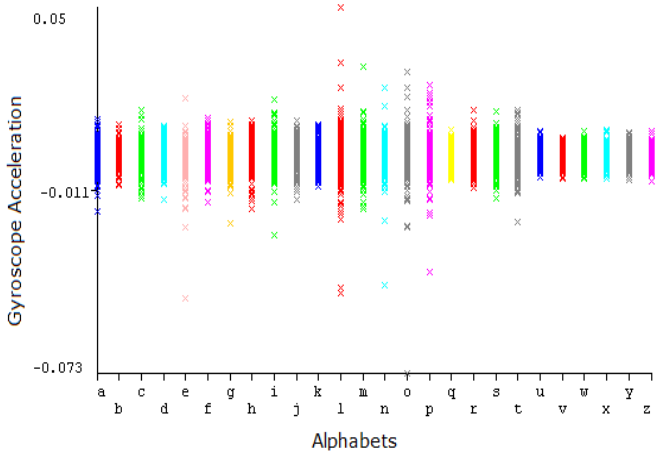

(b) Unique Accelerometer $A_y$ Dimension Pattern



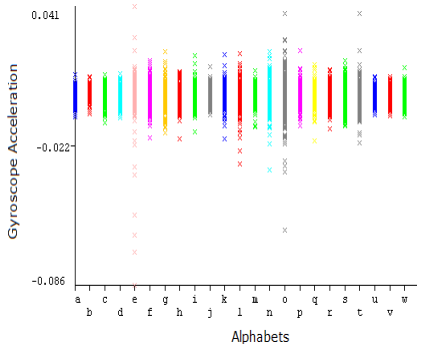

(c) Unique Accelerometer $A_z$ Dimension Pattern

Fig. 2: Unique pattern illustration Accelerometer showing the variation of readings on each keystroke.

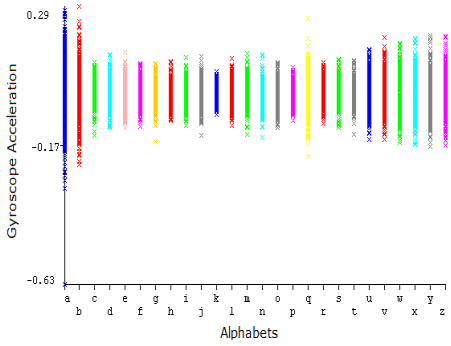
In the case of the gyroscope all combination scenarios, it has a negative impact. The gyroscope returns three-axis reading according to the gravitational pull along the x-axis, y-axis, and z-axis. In our particular case, there are no chances of rotation that is why it does not shows promising results as shown in Figures 3a, 3b, 3c.




(a) Gyroscope $G_x$ Dimension Pattern



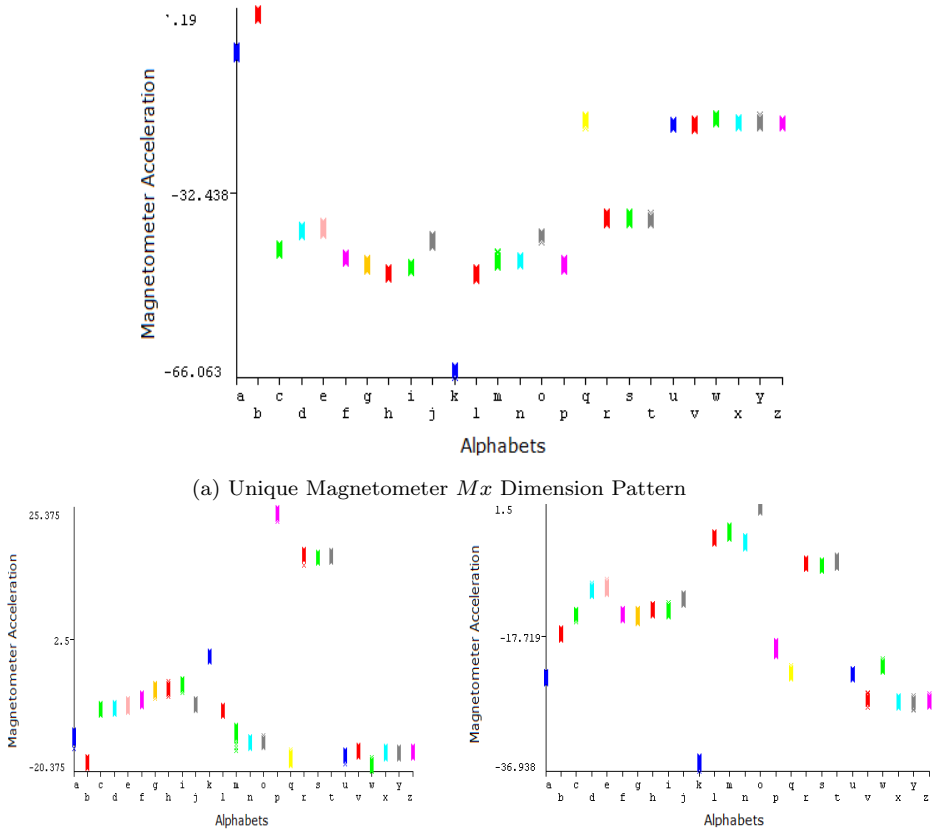

(b) Gyroscope $G_y$ Dimension Pattern




(c) Gyroscope $G_z$ Dimension Pattern

Fig. 3: Pattern illustration of Gyroscope showing the variation of readings on each keystroke.

The magnetometer sensors show unique behavior along the three axes in figure 4a, 4b, 4c. Magnetometer contributes to enhancing the inference rate when it is used with the accelerometer.




(a) Unique Magnetometer $Mx$ Dimension Pattern

(b) Unique Magnetometer $M_y$ Dimension Pattern

(c) Unique Magnetometer $M_z$ Dimension Pattern

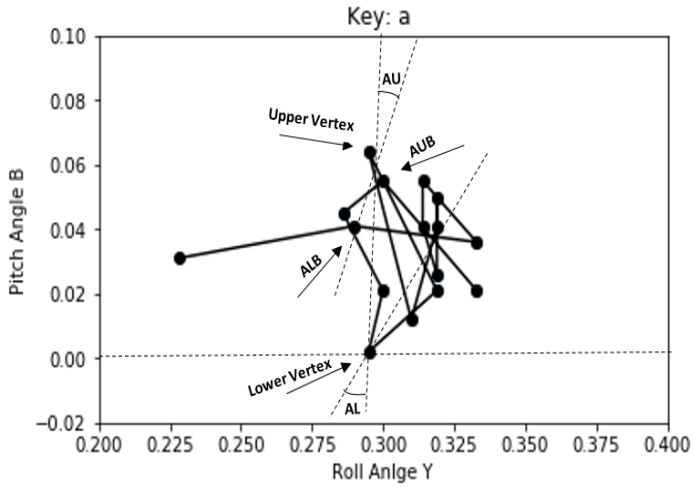
Fig. 4: Pattern illustration of Magnetometer showing the variation of readings on each keystroke.

## 4.2 Typical Pattern Analysis

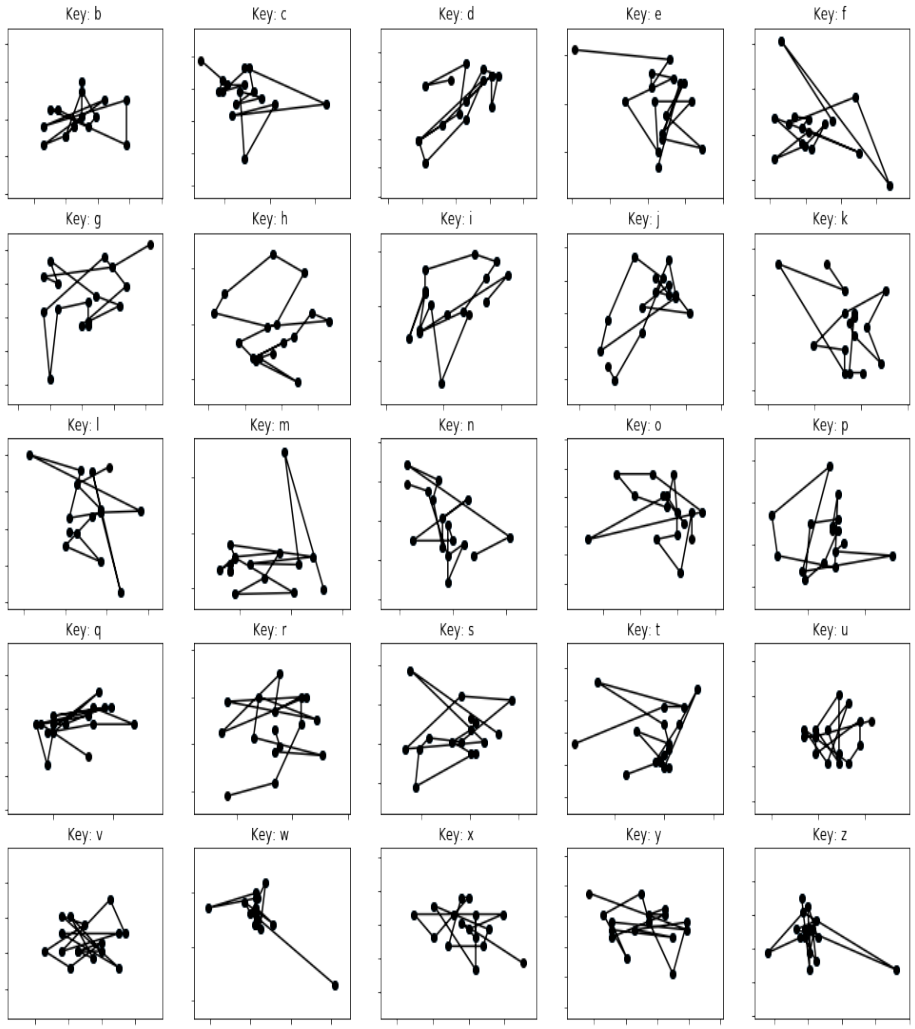
Keystrokes can be inferred using device orientation while typing on the smartphone screen. Typing a specific word produces a particular vibration in a specific direction which causes the change in the axis of smartphone orientation which can be analyzed using the accelerometer sensor. We extract typical patterns for each alphabet on a soft keyboard as shown in Figure 5b. For each event accelerometer data consists of three-axis $(\alpha_j, \beta_j, \gamma_j)$ , j = 1,..., n, where $\alpha_j$represents the azimath, $\beta_j$ represents the pitch, $\gamma_j$ represents the roll angles.

- $\beta$ : represents the rotation along x-axis. $\beta$ (pitch angle) produces variations in between the range of $[-180, 180]$.
- $\gamma$ : represents the rotation along the y-axis. $\gamma$ (roll angle) produces variations in between the range of $[-90, 90]$.

Device orientation depends only on two measures: $\beta$ and $\gamma$ therefore we drop the $\alpha$ also known as azimuth feature. We estimate each tap event using peak to average as typing on smartphone causes vibration which produces a peak and then it goes down to the average.
Figure 5b shows that each alphabet represents a unique pattern. We observe that the angle between lobes can be used to distinguish the typed alphabet from others. $AU$ is the angle between an upper vertex and $AL$ is the angle between the lower vertex. $AUB$ illustrates the angle of the upper bisector and $ALB$ represents the angle of the lower bisector. This analysis helps to understand the flow of vibration when a particular keyboard is pressed along all axis of each sensor. It is seen that the z-axis of the accelerometer sensor does not contribute to understanding the flow of vibration. Each alphabet, when pressed continuously, produces repetitive behavior.




(a) Pitch angle of alphabet A




(b) Pitch angle of all other alphabets

Fig. 5: Pattern analysis of different angles when alphabets are pressed.

### 4.3 Results and Discussion

To evaluate the performance of *AlphaLogger*, we use Accuracy, Precision, Recall, F-Value, and Area Under Curve (AUC) as evaluation metrics (Hall et al. 2009). Accuracy represents the overall correctly predicted instances, and is given as $\frac{TP+TN}{TP+TN+FP+FN}$, where $TP$ represents the true positive which in our case are the number of examples predicted true that are actually true, $TN$ number of examples predicted negative that are actually negative, $FP$ represents number of examples predicted positive that are actually negative and $FN$ represents the false negatives. Since our dataset contains imbalanced class representation, we also

considered Precision, Recall, and F-Value to determine the performance of our proposed methodology. Where Precision and Recall are calculated using $\frac{TP}{TP+FP}$ and $\frac{TP}{TP+FN}$ respectively, While F-Value is the weighted average of the Precision and Recall and is given as $2 \times \frac{Precision \times Recall}{Precision+Recall}$. Moreover, AUC represents the degree or measure of separability. The AUC shows how sensitivity and specificity vary at every possible threshold.

The result of our proposed keystroke inferring techniques is presented in Table 1 illustrating shows the predicted F-Value. *AlphaLogger* achieves an accuracy of over 90% on a dataset containing magnetometer and accelerometer readings, 86.5% when accelerometer, gyroscope, and magnetometer sensor readings are used, 59% when the accelerometer is used in combination with gyroscope and 60.3% when only the accelerometer readings are used. The keys with the highest inference accuracy are alphabets ***a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, r, s, t, w*** and the lowest inference accuracy are alphabets ***v, x, y, z***. Lowest inference alphabet ***x, z*** are located on a corner of the soft keyboard while ***u*** and ***y*** get confused with each other. The reason behind this confusion is that participants typed alphabets while holding mobile with both hands and pressing keys with the thumb. The location of ***u*** and ***y*** is difficult to press with thumb as it can be seen in Figure 5b that they are confused with each other. It is common for the user to mistakenly press ***u*** while pressing ***y***. Highest inference alphabets are consistent as shown in Figures 2, 3 and 4. We observe that physical location of the alphabets decreases inference accuracy.

Table 1 describes the accelerometer results when a model is build using readings of accelerometer alone. We observe that using the only accelerometer does not perform well. A major drawback of using an accelerometer alone is that the accelerometer sensor gets the reading against each hit on a character. The z-axis of the accelerometer does not contribute to inferring keystrokes. It might help when the device rotates along the z-axis. We chose to use it in combination with other sensors as explained in Table 1 which resulted in increasing the overall performance of the model.

Accelerometer and gyroscope combination in Table 1 describes the result when a model is build using readings of accelerometer and gyroscope. We observe that using an accelerometer reading with a gyroscope does not perform well. It harms the overall performance of the model as it produces constant readings which confuse the machine learning model in keystroke inference. A major drawback of using gyroscope with the accelerometer is that the gyroscope sensor provides orientation information of the device in three axes. In this case, the device is not rotating that is why it is useless in this particular scenario.

Accelerometer and magnetometer combination in Table 1 describes the result when a model is build using readings of accelerometer and magnetometer. We observe that using an accelerometer reading with a magnetometer performs well. A major benefit of using a magnetometer with the accelerometer is that the magnetometer sensor lets you measure the magnetic field around the participant.

Accelerometer, gyroscope and magnetometer combination in Table 1 describes the result when a model is build using readings of accelerometer, gyroscope, and magnetometer. We observe that using an accelerometer and magnetometer reading with a gyroscope does not perform well. It harms the overall performance of the model. A major drawback of using gyroscope with the accelerometer is that the

gyroscope sensor provides information about the rotation of the device in three axes. In this case, the device is not rotating that is why it is not helpful in this particular scenario.

Table 1: F-Value of Inference Performance of Different Combination of Sensors.

| **Character** | **Accelerometer** | **Accelerometer+ Gyroscope** | **Accelerometer+Gyroscope +Magnetometer** | **Accelerometer+ Magnetometer** |
|---|---|---|---|---|
| **a** | 0.206 | 0.223 | 1.000 | 1.000 |
| **b** | 0.227 | 0.183 | 1.000 | 1.000 |
| **c** | 0.798 | 0.849 | 0.999 | 0.999 |
| **d** | 0.967 | 0.987 | 0.992 | 0.990 |
| **e** | 0.961 | 0.982 | 0.992 | 0.990 |
| **f** | 0.738 | 0.750 | 0.999 | 0.999 |
| **g** | 0.531 | 0.457 | 1.000 | 0.999 |
| **h** | 0.966 | 0.971 | 0.976 | 0.986 |
| **i** | 0.918 | 0.934 | 0.976 | 0.934 |
| **j** | 0.999 | 0.998 | 1.000 | 1.000 |
| **k** | 0.766 | 0.782 | 1.000 | 1.000 |
| **l** | 0.942 | 0.948 | 0.997 | 1.000 |
| **m** | 0.966 | 0.955 | 0.998 | 0.999 |
| **n** | 0.887 | 0.864 | 0.997 | 0.999 |
| **o** | 0.586 | 0.593 | 0.998 | 1.000 |
| **p** | 0.466 | 0.445 | 1.000 | 1.000 |
| **q** | 0.158 | 0.120 | 0.616 | 0.821 |
| **r** | 0.528 | 0.477 | 0.968 | 0.978 |
| **s** | 0.977 | 0.953 | 0.969 | 0.979 |
| **t** | 0.821 | 0.775 | 0.999 | 0.999 |
| **u** | 0.302 | 0.279 | 0.714 | 0.865 |
| **v** | 0.253 | 0.199 | 0.445 | 0.600 |
| **w** | 0.184 | 0.160 | 0.926 | 0.977 |
| **x** | 0.191 | 0.169 | 0.31 | 0.414 |
| **y** | 0.127 | 0.148 | 0.315 | 0.464 |
| **z** | 0.2 | 0.136 | 0.291 | 0.391 |
| **Average** | **0.603** | **0.590** | **0.865** | **0.902** |

Figure 6 shows average results of all the combination of sensors. We found that results using the reading of accelerometer and magnetometer are better than all other combination. Gyroscope has negative impact on the inference process.

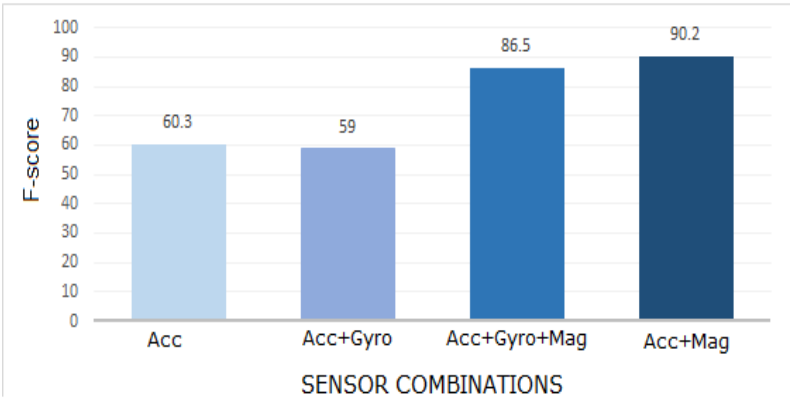



Fig. 6: Result Comparison of Different Combination of Sensors.

Figure 7 illustrates the confusion matrix of our machine learning model. We see that keystrokes in the smartphone keyboard's last row of alphabet especially alphabets in the left corner of the keyboard produce confused vibrations while typing as shown in Figure 7 as well as in Figure 5b showing typical patterns of each alphabet. This is due to the location of the alphabets on a soft keyboard. Only 4 keystrokes (x,y,z, and v) are the alphabets inferred with low accuracy while all other alphabets are inferred with high accuracy.

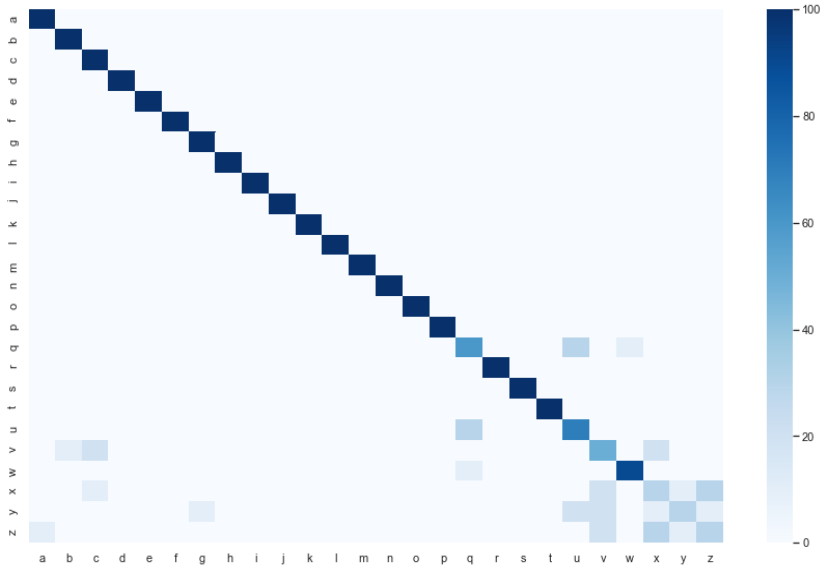



Fig. 7: 90.2% keystrokes were correctly inferred using Accelerometer in combination with Magnetometer

### 4.4 Comparative analysis of classification algorithms

Multiple machine learning classifiers are trained: J48, kNN, SMO, MLP, Adaboost and Ensemble voting method for supervised classification to infer keystroke. We investigate if model could learn the boundary between twenty six alphabet classes provided using motion data. For parameter tuning, such as $K$ in Equation 3, by using *Radial Basis Function (RBF) kernel AlphaLogger* got the result of 88.5% on the training data of accelereomter and magnetometer. We also train this model on different kernel functions: *puk kernel, normalized puk, poly kernel* and different combination of *alpha and omega*. When we tune this kernel function to the *puk kernel*, alpha and omega to $1.0, 1.0$ respectively, we observe an improvement in accuracy. For kNN, two most important parameters are *distance function* and *number of nearest neighbor*. We chose Euclidean distance as distance function and $K = n$ nearest neighbors. We tune the value of k and determine that the kNN is providing best performance when it is used with k=3 neighbors. For decision tree (J48), we use the confidence factor of 0.1 along with pruning parameter set to the "false". This model is then used to infer the keystrokes. In voting meta classifier, we use the above setting of the algorithms and chose kNN as the base classifier, given the highest priority among all other classifiers. Table 2 presents the comparative results of all algorithms and ensemble voting of five machine learning algorithms combined with averaging or voting. The result in Table 2 shows the best accuracy of 90.2% using voting algorithm.

Table 2: Comparison of Different Machine Learning Algorithm Using Combination of Accelerometer and Magnetometer

| **Algorithm** | **Precision** | **Recall(%)** | **F-Value** | **Accuracy** | **AUC** |
|---|---|---|---|---|---|
| **J48** | 0.888 | 0.888 | 0.888 | 0.888 | 0.967 |
| **MLP** | 0.891 | 0.891 | 0.891 | 0.890 | 0.964 |
| **SMO** | 0.889 | 0.880 | 0.871 | 0.880 | 0.991 |
| **Adaboost** | 0.889 | 0.880 | 0.871 | 0.884 | 0.991 |
| **Voting** | **0.902** | **0.902** | **0.902** | 0.902 | **0.95** |

Figure 8 depicts the result comparison of the *AlphaLogger* using the selected evaluation metrics: Precision, Recall, F-Value, Accuracy, and AUC. The voting algorithm outperforms all other machine learning algorithms in all evaluation metrics. All other algorithms perform equally in terms of all evaluation metrics. The only Voting algorithm performs better than all classifiers because it makes the decision of keystroke inference based on the votes provided by sub classifier: J48, kNN, SMO and MLP. A decision is made on the majority basis. If a keystroke is predicted same by majority classifiers then the voting algorithm returns that decision. The Voting achieves the overall best accuracy of 90.2%.

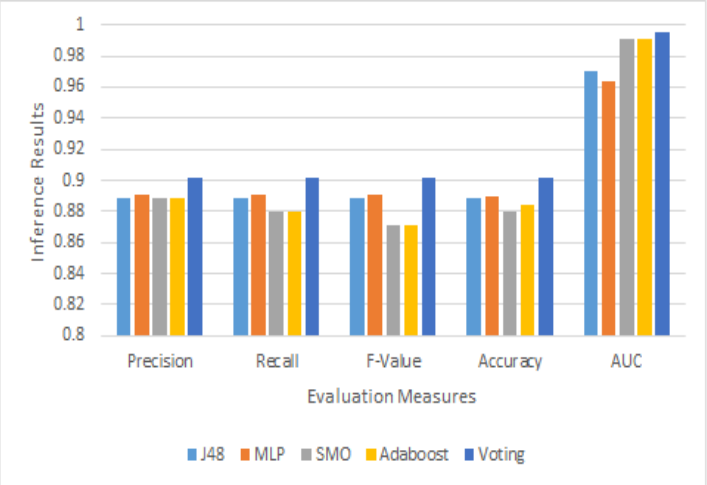



Fig. 8: 90.2% keystrokes were correctly inferred using Accelerometer in combination with Magnetometer

Table 3 depicts the result comparison with state of the art paper (Ping et al. 2015), in which authors report the highest accuracy of 36.2% using ensemble of 4 algorithms: Simple logistic, random forest (RF), SMO and kNN. There are some similarities in the results of both (Ping et al. 2015) and *AlphaLogger*, such as the keys on the smartphone screen located at the corner of the screen are difficult to classify as shown in Table 3 (like x, y, z, and v). In both studies the accuracy of these keys are quite low . By using sensor fusion technique *AlphaLogger* achieves the highest accuracy of 90.2%.

Table 3: Accuracy Comparison of *AlphaLogger* with TextLogger (Ping et al. 2015)

| Character | AlphaLogger | TextLogger |
|---|---|---|
| **a** | **100.0** | 67.9 |
| **b** | **100.0** | 8.77 |
| **c** | **99.0** | 31.25 |
| **d** | **99.0** | 20.31 |
| **e** | **99.0** | 76.03 |
| **f** | **99.9** | 10.71 |
| **g** | **99.9** | 3.64 |
| **h** | **98.6** | 15.38 |
| **i** | **93.4** | 51.24 |
| **j** | **100.0** | 16 |
| **k** | **100.0** | 24.07 |
| **l** | **100.0** | 33.8 |
| **m** | **99.9** | 29.55 |
| **n** | **99.9** | 36.51 |
| **o** | **100.0** | 30.68 |
| **p** | **100.0** | 43.55 |
| **q** | **82.1** | 57.63 |
| **r** | **97.8** | 18.42 |
| **s** | **97.9** | 35.53 |
| **t** | **99.9** | 43.94 |
| **u** | **86.5** | 38.27 |
| **v** | **60.0** | 34.92 |
| **w** | **97.7** | 24.53 |
| **x** | **41.4** | 22.03 |
| **y** | **46.4** | 43.75 |
| **z** | **39.1** | 16.33 |

## 5 Conclusion and Future Work

In this paper, we investigate the use of hardware sensors (such as accelerometer, gyroscope, and magnetometer) to infer the typed characters on the smartphone soft keyboard. Although side-channel attacks have been discussed widely in recent studies, the problem of inferring cross-application keystrokes has, so far, been overlooked. We developed an Android-based application *AlphaLogger* that is capable of inferring character while instant writing in any application. An extensive evaluation showed that the *AlphaLogger*, we have used works better when sensors are used in combination with the magnetometer sensor resulting in an accuracy of 90.2%. We got the promising results as compared to the proceeding work and show that data leakage from other applications can also be sniffed. We believe that *Alphalogger* is significant for inferring important information on a smartphone. In the future, we plan to use natural language processing (NLP) and text mining techniques to extract more sensitive information from the inferred text.

## References

Ali R, Pal AK, Kumari S, Sangaiah AK, Li X, Wu F (2018) An enhanced three factor based authentication protocol using wireless medical sensor networks for healthcare monitoring. Journal of Ambient Intelligence and Humanized Computing pp 1–22

Altman NS (1992) An introduction to kernel and nearest-neighbor nonparametric regression. The American Statistician 46(3):175–185

Aviv AJ, Sapp B, Blaze M, Smith JM (2012) Practicality of accelerometer side channels on smartphones. In: Proceedings of the 28th Annual Computer Security Applications Conference, New York, pp 41–50

Cai L, Chen H (2011) Touchlogger: Inferring keystrokes on touch screen from smartphone motion. In: Proceedings of the 6th USENIX conference on Hot topics in security, p 9

Cai L, Machiraju S, Chen H (2009) Defending against sensor-sniffing attacks on mobile phones. In: Proceedings of the 1st ACM workshop on Networking, systems, and applications for mobile handhelds, Barcelona, pp 31–36

Cook DJ (2010) Learning Setting-Generalized activity models for smart spaces. IEEE intelligent systems 2010(99):1

Deb S, Yang YO, Chua MCH, Tian J (2020) Gait identification using a new time-warped similarity metric based on smartphone inertial signals. Journal of Ambient Intelligence and Humanized Computing pp 1–13

Dietterich TG (2000) Ensemble methods in machine learning. In: International workshop on multiple classifier systems, Italy, Springer, pp 1–15

Foo Kune D, Kim Y (2010) Timing attacks on pin input devices. In: Proceedings of the 17th ACM conference on Computer and communications security, Chicago Illinois USA, pp 678–680

Graczyk M, Lasota T, Trawiński B, Trawiński K (2010) Comparison of bagging, boosting and stacking ensembles applied to real estate appraisal. In: Asian conference on intelligent information and database systems, Yogyakarta Indonesia, Springer, pp 340–350

Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH (2009) The weka data mining software: an update. ACM SIGKDD explorations newsletter 11(1):10–18

Hussain M, Al-Haiqi A, Zaidan A, Zaidan B, Kiah MM, Anuar NB, Abdulnabi M (2016) The rise of keyloggers on smartphones: A survey and insight into motion-based tap inference attacks. Pervasive and Mobile Computing 25:1–25

Krause A, Ihmig M, Rankin E, Leong D, Gupta S, Siewiorek D, Smailagic A, Deisher M, Sengupta U (2005) Trading off prediction accuracy and power consumption for context-aware wearable computing. In: Ninth IEEE International Symposium on Wearable Computers (ISWC'05), Osaka, IEEE, pp 20–26

Kucukyilmaz T, Cambazoglu BB, Aykanat C, Can F (2008) Chat mining: Predicting user and message attributes in computer-mediated communication. Information Processing & Management 44(4):1448–1466

Kuppusamy K (2019) PassContext and PassActions: transforming authentication into multi-dimensional contextual and interaction sequences. Journal of Ambient Intelligence and Humanized Computing pp 1–28

Kwapisz JR, Weiss GM, Moore SA (2011) Activity recognition using cell phone accelerometers. ACM SigKDD Explorations Newsletter 12(2):74–82

Lanette S, Mazmanian M (2018) The smartphone" addiction" narrative is compelling, but largely unfounded. In: Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems, Montreal, Canada, pp 1–6

Owusu E, Han J, Das S, Perrig A, Zhang J (2012) ACCessory: password inference using accelerometers on smartphones. In: Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications, San Diego California, pp 1–6

Ping D, Sun X, Mao B (2015) Textlogger: inferring longer inputs on touch screen using motion sensors. In: Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks, New York, pp 1–12

Platt J (1998) Sequential minimal optimization: A fast algorithm for training support vector machines

Ruan O, Wang Q, Wang Z (2019) Provably leakage-resilient three-party password-based authenticated key exchange. Journal of Ambient Intelligence and Humanized Computing 10(1):163–173

Shumailov I, Simon L, Yan J, Anderson R (2019) Hearing your touch: A new acoustic side channel on smartphones. arXiv preprint arXiv:190311137

Song R, Song Y, Gao S, Xiao B, Hu A (2018) I know what you type: Leaking user privacy via novel Frequency-Based Side-Channel attacks. In: 2018 IEEE Global Communications Conference (GLOBECOM), Abu Dhabi, IEEE, pp 1–6

Tang B, Wang Z, Wang R, Zhao L, Wang L (2018) Niffler: A Context-Aware and User-Independent Side-Channel. Wireless Communications and Mobile Computing 2018

Tang G, Pei J, Luk WS (2014) Email mining: tasks, common techniques, and tools. Knowledge and Information Systems 41(1):1–31

Voicu RA, Dobre C, Bajenaru L, Ciobanu RI (2019) Human physical activity recognition using smartphone sensors. Sensors 19(3):458

Vuagnoux M, Pasini S (2009) Compromising electromagnetic emanations of wired and wireless keyboards. In: Proceedings of the 18th conference on USENIX security symposium, Montreal, pp 1–16

Wang Y, Cai W, Gu T, Shao W (2019) Your eyes reveal your secrets: An eye movement based password inference on smartphone. IEEE Transactions on Mobile Computing pp 1–1

Xu D, Chen J, Liu Q (2019) Provably secure anonymous three-factor authentication scheme for multi-server environments. Journal of Ambient Intelligence and Humanized Computing 10(2):611–627

Xu N, Zhang F, Luo Y, Jia W, Xuan D, Teng J (2009) Stealthy video capturer: a new video-based spyware in 3g smartphones. In: Proceedings of the second ACM conference on Wireless network security, Zurich Switzerland, pp 69–78

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

Zhuang L, Zhou F, Tygar JD (2009) Keyboard acoustic emanations revisited. ACM Transactions on Information and System Security (TISSEC) 13(1):1–26