Computers & Security 121 (2022) 102837

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

Keypad entry inference with sensor fusion from mobile and smart wearables

Yuanzhen Liu[a,1], Umair Mujtaba Qureshi[b], Gerhard Petrus Hancke[a,*]

[a] Department of Computer Science, City University of Hong Kong, Tat Chee Avenue, Kowloon, Hong Kong
[b] School of Professional Education and Executive Development, College of Professional & Continuing Education, The Hong Kong Polytechnic University, Hong Kong

ARTICLE INFO

Article history:
Received 29 November 2021
Revised 18 June 2022
Accepted 12 July 2022
Available online 20 July 2022

Keywords:
Sensors
Pin entry devices
Side-channel attack
Payment
PIN recovery

ABSTRACT

This paper investigates the threat of password leakage from sensors on mobile phones and smart wearables. We investigate attack methods for recovering a short random number entered on PEDs using a combination of data collected by from microphone, accelerometer and gyroscope in devices on the person entering the number. We take into consideration features based on keypress sounds, from either phone and/or smart watch, in addition to hand movement acceleration and angular velocity captured by the smart watch. We used the fusion features from these three sensor sources to train a Neural Network to recover key entries on a keypad. Our method based on linear acceleration and angular velocity, with acoustic data used to segment individual keystrokes, showed an improved 74% probability to predict a single keystroke in one guess and 52% for a 6-digit PIN in three guesses.

1. Introduction

We are living in a digital age. Over 70% of the world's population is well exposed to digital gadgets, such as smart phones and smart wearables (e.g., smart watches, Fitbit and tags). The extensive use of smart gadgets in daily life has been exploited in different ways by companies and market stake holders, especially by the marketing agencies that gather or record our personal data via these gadgets to have useful insight into our preferences for advertising their products. One major concern in this context is the security of the personal information of a user who owns these gadgets. The most critical information is a numeric password or Personal Identification Number (PIN), which is one of the most widely used methods for user authentication. It is very common for people to have smart gadgets, such as smart phones and smart watches, while they enter a password or PIN on computers, laptops, tabs, customer kiosks or ATM machines. Entering a 6 or 8 digit PIN via a PIN Entry Device (PEDs) is shown in Fig. 1.

The Payment Card Industry (PCI) Standard Council defined the standards of security and testing requirements for certification of devices used for PIN entry for payment and transactions (PCI, 2011). One of the most important requirements for PEDs is that there should not be a feasible way to infer the entered digits on PEDs by recording and analysing the sound, electromagnetic emissions, power consumption or other external information. However, how strictly these standards are implemented is still a concern. In fact, it is observed that the devices that are used to input the PIN have security flaws (de Souza Faria and Kim, 2019), which result in information leakage. To guarantee information security, many researchers studied PINs Entry Devices (PEDs), including strengthening the security of PEDs and attacks on PEDs. For example, tampering and PIN/card details logging (Drimer et al., 2008), card wedge allowing transaction approval with no PIN(Murdoch et al., 2010), ineffective random number generation by PEDs for cryptographic functions(Bond et al., 2014). Except for the internal security issues, the unintended emission, e.g. acoustic emanation may also cause information leakage. This has allowed researchers to design side-channel attacks on PEDs (Joy Persial et al., 2011; Qureshi et al., 2018). The reference for attacks on PEDs are the work done by researchers who have devised attacks on PC keyboards by using features extracted from acoustic side-channels to train machine learning models to infer keystrokes, e.g. using frequency differences features to train Neural Networks to predict keystrokes (Asonov and Agrawal, 2004), or using cepstrum features to train K-mean models to infer keystrokes (Zhuang et al., 2009), using time difference of arrival (TDoA) to locate the key position (Zhu et al., 2014). For attacks on PEDs, researchers tried key inference on PEDs by using the acceleration data in smart watches (Liu et al., 2015) and key inference

* Corresponding author.
E-mail addresses: yuanzhliu3-c@my.cityu.edu.hk (Y. Liu), umair.qureshi@cpce-polyu.edu.hk (U.M. Qureshi), gp.hancke@cityu.edu.hk (G.P. Hancke).
[1] [orcid=0000-0002-5888-385X]

https://doi.org/10.1016/j.cose.2022.102837
0167-4048/

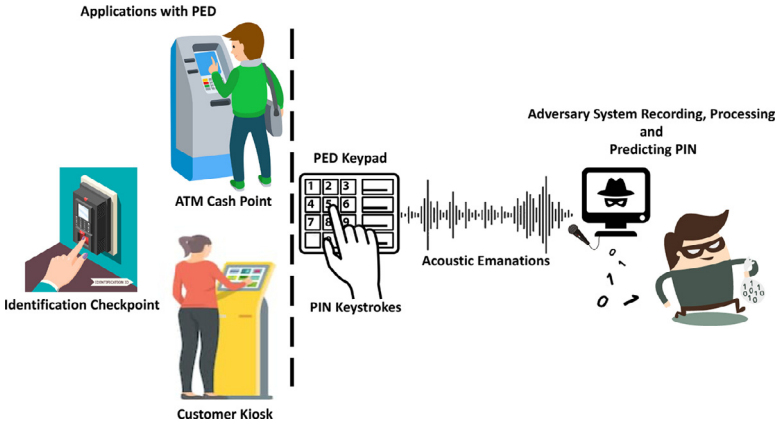

Fig. 1. PIN Attack: Acoustic emanations generated from PED keystrokes are recorded and processed to predict a PIN key by an adversary (Panda et al., 2020).

on touch screens by using the acceleration data in mobile phones (Owusu et al., 2012). From the literature, it is observed that by large, researchers have focus on single feature based key inference on PEDs i.e., they only tried using one feature to infer keystrokes, e.g., the acoustic feature or the acceleration feature, though it is possible to use multiple features to perform classification (Teh et al., 2016).

Therefore, the novelty of this paper is twofold: 1) We have attempted to infer a key in the scenario where there are two smart gadgets near a PED, out of which one is stationary while another is moving. 2) We perform key inference using the fusion of the acoustics emanations generated by the keystrokes of the PED and the motion data generated by the smart gadgets. The use of two smart gadgets, where sensor data is gathered of both and synchronised, makes the problem of key inference interesting yet very challenging. In this regard, in paper(LIU et al., 2021), we performed an evaluation of differentiating keys on PEDs by using acoustic emanations and motion data. The acoustic data that we used to calculate TDoA were collected from two smart gadgets, i.e., a mobile phone and smart watch. While recording the data, the mobile phone was static, but the smart watch was moving. In addition, the movement of the smart watch provided us with linear acceleration and angular acceleration datum from accelerometer sensor and gyroscope sensor of the smart watch. This allows us to work with multiple features, which has many advantages. Firstly, using more effective features can raise the accuracy of machine learning models (Alpaydin, 2020). Secondly, if one feature is not available, we can still perform classification. In this paper, therefore set out to evaluate which features from mobile and smart watch could be best combined to do keystroke inference. We consider TDoA (watch and phone being the reference devices), linear acceleration and angular velocity. We then introduce a new context-free side-channel attack method on PEDs using sensor data from both a mobile phone and a smart watch and a machine-learning model for keystroke inference. Thus, our main contribution is fourfold:

1. We used three sensor data sources for key inference on PED, i.e., acoustic emanation, linear acceleration and angular velocity, compared to existing work mostly focusing on a single sensor source.
2. We analyzed the impact of each of the features, using single features as well as different combinations of features to investigate the best approach to infer keystrokes on PED.
3. We use the fusion of acoustic (used for keystroke segmentation), linear acceleration and angular velocity features to train classifiers for inferring keystrokes.
4. Our accuracy of predicting single keystrokes can reach 74% in single keystroke prediction and 52% for guessing 6-digit PIN in three tries.

The rest of this paper is organized as follows: Section 2 shows the basic idea of our work. Section 3 describes our experiment, including the data collection procedure. 3.3 shows how we extract features from the raw data, how we select and train machine learning models and the result. Section 4 shows the related works of key inference and comparison between related works and our work. Section 5 draws the conclusion.

2. Keystroke inference & feature engineering

Our work is based on the following real-life scenario. A person walks towards an ATM machine or a kiosk to access his/her bank account or access any application via the ATM or Kiosk machine. The person has a cell phone in his pocket and a smart watch connected via Bluetooth on his left wrist. He reaches the ATM machine, plugs his ATM card into the ATM machine and starts to enter his PIN on an ATM PED with his left hand (the one with the smart watch). Furthermore, let us assume that there is a malicious application on the user's phone and smart watch. These applications have access to the watch and phone microphone as well as the sensor data of the watch. Malware and malicious applications are unfortunately known mobile security issues. Smart devices are now closely integrated and an attacker does not need to necessarily compromise both devices independently. For example, a fitness application installing on the mobile could prompt the user to also install the watch application. Currently, if you install an Android mobile application that has an associated Wear watch application, a notification appears on the watch offering the opportunity to install the watch application. When the user starts to enter his PIN on the ATM PED, the applications begins to record sound, linear acceleration and angular velocity. Once the user completes his PIN entry, the watch application shares its date with the phone application, which uploads all the data to the cloud via WiFi or mobile data for further for analysis. This sensor data can be processed to infer the PIN, thus causing information leakage. PIN is usually 4 to 8 digits long. In this scenario, we shall first establish the principal with which we are able to detect and distinguish between different digits of a PIN. In order to do PIN digit differentiation we infer keystrokes information provided by the acoustic and motion sensor to extract reliable features for PIN digit detection and differentiation. The subsequent section discusses the feature engineering from acoustics and motion sensors.

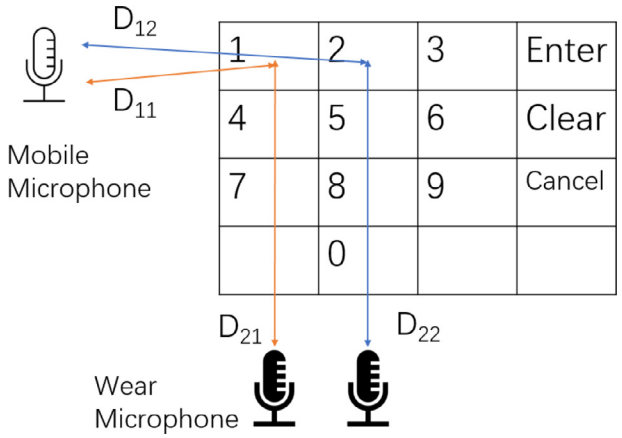

Fig. 2. Distance difference of key '1' and key '2' of mobile and wear microphones.

2.1. Acoustic feature

In this experiment, we used time difference of arrival (TDoA), which has been proved that can be used to estimate the signal delay(Azaria and Hertz, 1984) and effective in keystroke inference(Zhu et al., 2014), to distinguish sound sources. We used the microphones in the mobile phone and smart wearable to record acoustic data. The sampling rate of these microphones are 44100Hz. The propagation speed of sound in air is approximately 343m/s. Suppose we have one microphone and two sound sources, the distances between sound sources and microphone are d_1 and d_2, respectively. If these two sound sources generate a 1-frame sound wave concurrently, and we want to distinguish the sound sources in the sound data recorded by the microphone, d_1 and d_2 must satisfy Eq. (1):

$$|d_1 - d_2| > \frac{343m/s}{44.1kHz} \approx 0.78cm \tag{1}$$

However, when inputting the PIN, the keys will never be hit at the same time. Thus we cannot estimate the sound source locations by just calculating d1 – d2. Suppose we have two microphones, M_1 and M_2, and two sound sources S_1 and S_2, which generate two 1-frame sound waves. Let us mark the distance between M_1 and S_1 as D_{11}, M_1 and S_2 as D_{12}, M_2 and S_1 as D_{21}, M_2 and S_2 as D_{22}. Since the sound sources do not generate sound concurrently, if we want to distinguish the two sound sources by time differences of arrival, we cannot just calculate the arrival times of S_1 and S_2 in M_1 and M_2, we need to know the differences of distance. Let $D_1 = D_{11} - D_{12}$, $D_2 = D_{21} - D_{22}$, if we want to distinguish S_1 and S_2, these two differences of distance must satisfy

$$|D_1 - D_2| > 0.78cm \tag{2}$$

If the absolute value of D1 – D2 is less than 0.78cm, then the TDoA of S1 and S2 will be the same and we cannot decide the sound source.

In our experiment, we used microphones in a mobile phone and a smart wearable to record acoustic data. While entering keys, the microphone in the mobile phone is static, the microphone in the smart wearable is moving with the hands. The distance from the finger to the smart wearable will not change, no matter which key the user is pressing. Fig. 2 shows the distances between microphones and keys when pressing key '1' and key '2'.

In this situation, if we want to calculate the TDoA, the only variable is the distance between the mobile phone microphone and the keys. Suppose the distance between the mobile microphone and key 1 was D_{11}, the distance between the fingertip and the wearable microphone was D_{21}. We can get the distance between the mobile phone microphone and key '1' by calculating D_{21} - D_{11}. We can use the same method to calculate the distance differences of all keys. Suppose the distance between key '1' and the mobile phone microphone was 5 cm, the distance from fingertip to smart wearable microphone was 20 cm, distances between keys were 2.5 cm, we can calculate the distance differences of all keys and evaluate if the distance differences can be distinguished in the acoustic data. Table 1 shows distances differences of key '1' to key '9'.

Table 1
Mobile Phone and Wearable Distance differences of keys '1' to key '9' .

15.00	12.50	10.00
14.41	12.09	9.69
12.93	10.99	8.82

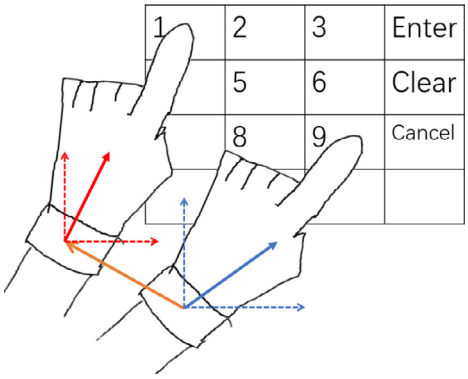

Fig. 3. Hand movement from key '9' to key '1'.

From Table 1, we can see that it is not possible to distinguish keys of row 1 and row 2 in the same column, e.g., key '1' and key '4', since the distance differences is less than 0.78 cm. Hence we need more data to make a precise classification. When collecting data, we recorded the sound, linear acceleration and angular velocity at the same time. The data collection details are explained in the next section.

2.2. Motion feature

Except for the microphone, the accelerometer and gyroscope in the smart wearable have potential to be used in the key inference. If we recover the movement during the key input using an accelerometer, we can know which key is pressed (Liu et al., 2015). The accelerometer can output a 3-dimensional vector, showing the acceleration value of the device during movement. The orange arrow in Fig. 3 shows the movement from key '9' to key '1'. If we apply integration twice to these acceleration values, we can roughly get the trace of the movement and use these traces to infer keystrokes. A gyroscope is another sensor in a smart wearable, which is used to record the rotation changes. It can show the angular velocity of the device in a 3-dimensional vector. When we move our hand from one key to another, the rotation of the smart wearable also changes. The blue arrows in Fig. 3 indicate the direction of smart wearable when pressing key '9' and the red arrow indicates the direction of smart wearable when pressing key '1'. If we integrated the angular velocity of the smart wearable, we can get the angular movement of the smart wearable and use this feature to infer keystrokes.

3. PIN inference experiment

In the last section we described the basic idea of using TDoA, linear acceleration and angular velocity to infer keystrokes on an ATM keyboard. Next we need to launch experiments to prove that this method can work in real life. The first step of the experiment is data collection.

3.1. Data collection

3.1.1. Hardware

We used 3 devices to collect data, a mobile phone, a smart watch connected with Bluetooth, and an ATM keyboard. The mobile phone's model is VIVO iQOO Neo, the operating system of this phone is Android 10. There are two microphones in this phone, one is on the top and the other is at the bottom. The smart watch's model is Ticwatch E2, there is one microphone on the left side of the watch, one accelerometer and one gyroscope inside the watch.

3.1.2. Software

We developed two Android applications to collect data, one was installed in the phone, the other was installed in the watch. In the phone application, there are three text labels and two buttons. The first label shows current status. The second label shows a 6-digit random number. The range is between 0 to 999,999. If the number is less than 100,000, zeros will be appended at the front of the number to keep the 6-digit format. The third label shows the number of collected samples. The two buttons are named 'Start' and 'Stop' respectively.

The watch application is fully controlled by the phone application, there is only one text label in the watch application, indicating the status. When we start collecting data, we first click the 'Start' button on the phone, then the phone begins to collect acoustic data and change the status label to 'Recording Started' and change the number in the random number label. At the same time, the phone sends a 'start' message to the watch. After received this message, the watch starts recording acoustic data and the timestamp of the start time. At the same time, the watch records linear acceleration data, angular velocity data and the timestamps of each data frame respectively. Concurrently, the watch changes its label to 'Recording Started'. If the "Stop" button was pressed, the phone will stop recording, change the status label to "Recording Stopped" and send a "stop" message to the watch.

The watch stops recording after receiving this message, then sends all the data including sound, linear acceleration and angular velocity data back to the phone and changes the label to "Recording Stopped'. After the phone has received data from the watch, it changes the status label to "Completed" and add one to the sample number and changes the sample number label to current number. Data is then transferred to database for further analysis.

3.1.3. Collection procedure

When collecting data, the mobile phone is located to the left side of the ATM keyboard, while the user wearing a smart watch on the left wrist is located in front of the keyboard. In this experiment, we were simulating a user using his left hand to input the PIN, with smart wearable on his left wrist and phone in the left pocket. Fig. 4 shows the positions of mobile phone, keyboard and user while collecting data.

We collected two data sets. The first data set includes 200 samples, the 6-digit numbers of these samples are the same, '010309'. This data set is used to test the feasibility of using TDoA, linear acceleration and angular velocity to infer keystrokes. The second data set includes 3000 6-digit PINs, which means 18,000 number key and 3000 "Enter" key samples are recorded. When collecting data, we first launch the application mentioned in the last section on

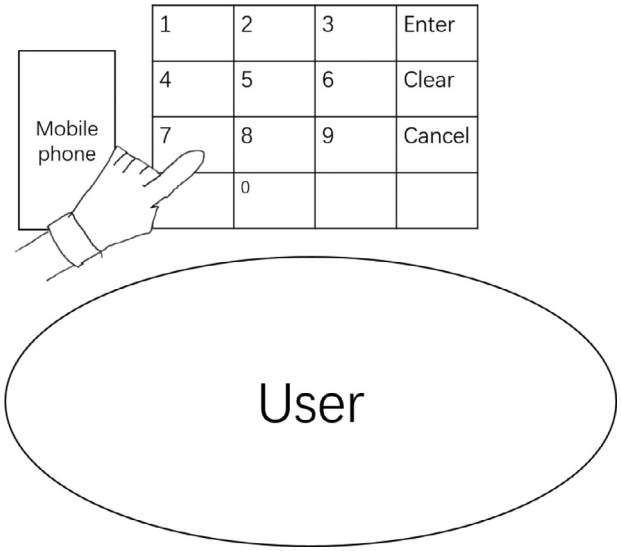

Fig. 4. Positions of mobile phone, keyboard and user while collecting data.

the mobile phone and the smart watch. Then we put the phone beside the ATM keyboard, wear the smart watch on the left wrist. Then we click the "Start" button. After we have noticed that both the phone and watch changed status labels to "Recording Started", we begin to input the 6-digit random number showed on the mobile phone on the ATM keyboard. After we finished inputting the random number, we press the "Enter" key on the keyboard. Then we click the "Stop" button on the phone and wait for data transmission from the watch to the phone, which is indicated by the status label.

We collected a total of 18,000 samples. The raw data in these samples cannot be used in key inference directly, we need to extract useful information from these data. The next section shows how we extract features from the raw data.

3.2. Feature extraction

After data collection, we have 18,000 samples of raw data, each sample includes 7 files: a text file records the 6-digit PIN and the recording start time of the phone and the watch, the acoustic data recorded by the phone and the watch, the linear acceleration and angular velocity data as well as the timestamps of each frame of these two kinds of data. These data cannot be directly used to infer keystrokes. We need to pre-process these data to extract useful information.

3.2.1. Acoustic data segmentation

As Section 2.1 explained, we can use TDoA to distinguish the keystrokes on an ATM keyboard. To calculate TDoA of keystroke sounds, we need to know which parts of the acoustic data are generated by the keystrokes. First we set all the acoustic data to their absolute value, then we used the "findpeak" function in MATLAB to detect the maximum amplitudes of each keystroke sound. The minimum distance of each peak parameter of this function is set to 5500 frames and the minimum peak value is set to 0.03. Fig. 5 shows a result of peak detection.

From Fig. 5 we can observe that the absolute values of release peaks are generally larger than absolute values of press peaks. To further prove this assumption, we made a statistic of all the mobile phone acoustic data of channel 2 including 21,266 keystrokes. The result shows that in 775 keystroke samples, the absolute value of press peak is larger than release peak; in 20,491 samples, the

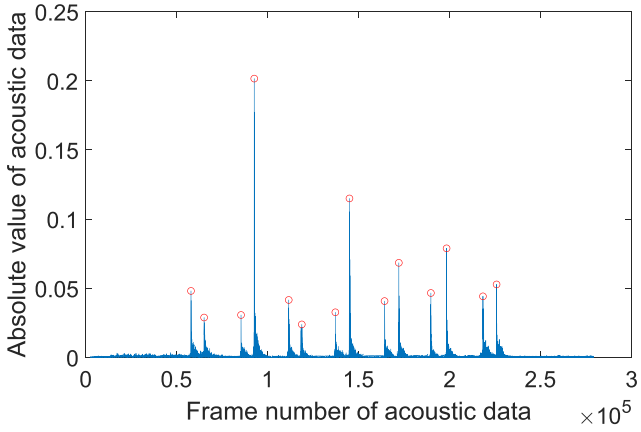

Fig. 5. Peak detection result of a soundtrack.

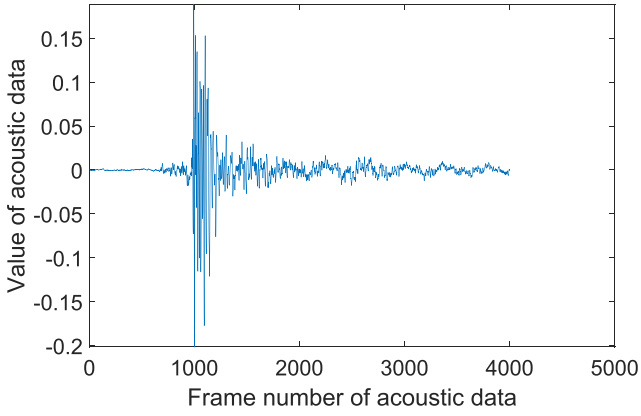

Fig. 6. Frames from peak minus 1000 to peak plus 3000 of release sound.

absolute value of release peak is larger than press peak. Hence we decided to use the release peak to estimate TDoA.

After the peak detection, we need to extract data frames, which include the sound of keystrokes. From our results, we found that using the peak minus 1000 and peak plus 3000 as the front and rear index can cover a peak or release acoustic wave. Fig. 6 shows the frames from peak minus 1000 to peak plus 3000 of a release sound wave.

3.2.2. Time synchronization and TDoA estimation

As Section 3.1.2 mentioned, we used both the phone and the watch to record the keystroke sound. When using multiple devices, time synchronization cannot be evaded. From Table 1 we can see that the theoretical TDoA value is less than 1 millisecond. Thus our experiment has a strict requirement for time synchronization. However, the accuracy of timestamps in Android system is 1 millisecond, thus we cannot use the timestamp to align sound data in the phone and the watch. Instead, we can use the “Enter” key to synchronize time. When entering a PIN on an ATM keyboard, the last key that we press is always “Enter”, this means we can use the “Enter” key as a landmark, then use the distances to the “Enter” key to locate other 6 keystroke sounds.

Let L_M and L_W denote the location of the last peak value, in acoustic data of mobile phone and smart wearable respectively. In our experiment, the last key is always the 'Enter key'. Let $M_{n,m}$, $W_{n,m}$ denotes the acoustic data from location n to location m, recorded by mobile phone and wearable respectively. 'GCC” means generalized cross-correlation. We can get the TDoA of the 'Enter' key by this equation:

$$T = GCC\{M_{L_M-1000,L_M+3000}, W_{L_W-1000,L_W+3000}\} \quad (3)$$

After we get T, we can start calculating the TDoA of other keys. Let L_{Mn} denotes the location of the n^{th} peak value in acoustic data recorded by mobile phone. Let D_{Mn} denotes the distance between the location of n^{th} peak value and the last peak value:

$$D_{Mn} = L_M - L_{Mn} \quad (4)$$

After we get D_{Mn}, We can get the keystroke sound clip of nth key recorded by both phone and wearable and calculate the TDoA of nth key. Let T_n denotes the TDoA value of the nth key:

$$T_n = GCC\{M_{L_M-1000-D_{Mn},\ L_M+3000-D_{Mn}},\ W_{L_W-1000-D_{Mn}-T,\ L_W+3000-D_{Mn}-T}\} \quad (5)$$

In our experiment, we calculated both cross-correlation and generalized cross-correlation. We used the ‘gccphat’ function in MATLAB to calculated the generalized cross-correlation of the two sound clips. However, since the qualities of sound records were not always perfect, e.g., in Fig. 6 we can see the noise in the data, the generalized cross-correlation sometimes would give an impossible result, e.g., more than a 1000 frames delay. Thus we also used cross-correlation to estimate TDoA. To calculate the cross-correlation of two sound clips, we used the ‘xcorr’ function in MATLAB, the maximum lag was set to 100. The function returns the correlation value and lag value, we select the lag value, which has the largest correlation value as the final result of TDoA of two sound clips.

From the above processes, we can estimate the TDoA of all number keys. Though the values are not the real value, since the TDoA of the ‘Enter’ keystroke sound is manually set to zero, we can still use this TDoA to distinguish keystrokes. Suppose the real TDoA of the ‘Enter’ key is T, the TDoA of the first key is T_1, The TDoA of the second key is T_2, since the TDoA of the ‘Enter’ key is set to zero, the real TDoA of the first key should be T_1 - T, the real TDoA of th second key should be T_2 - T. The difference of these two TDoA’s is (T_1 - T) – (T_2 - T), the T actually does not affect the difference value. Thus we can use these TDoA values to distinguish keystrokes.

To prove the feasibility of this method, we tested this method using the 200-sample data set mentioned in Section 3.1.3. We applied this method to the collected samples and made a statistic. The result shows that some keys can be distinguished: Most TDoA values of Key ’0’ are in the range (-14,-13), the TDoA of Key ’3’ is in the range (-2,-1), however, most TDoA’s of Key ’1’ and Key ’9’ are in the range (-7,-6). Fig. 7 shows the distribution of these keystrokes’ TDoA’s.

3.2.3. Linear acceleration

Section 3.2.2 shows that it is possible to infer some keystrokes by the TDoA. However, we cannot figure out key ’1’ and key ’9’. Thus we need to gain more data to distinguish keystrokes. One possible data source is the accelerometer. A data frame in the acceleration data has 3 components, X axis, Y axis and Z axis. The X and Y axis record our horizontal movements while the Z axis records our vertical movements. As Section 3.1.2 mentioned, we recorded acceleration data and the timestamp when recording the data in each sample. By our observation, we found that the sampling rate of the accelerometer is non-uniform. To make this data easier to use, we need to interpolate them. In our experiment, we used the cubic spline function in MATLAB to interpolate the data. Fig. 8 shows the Z axis data before and after interpolation. From Fig. 8 we can see that the shape of data hardly change after the interpolation.

When inputting the PIN on an ATM keyboard, we first move a finger to above the key, then press the key and the key generates a press sound, which is recorded by the microphone in watch. After that we release the key and move to another key. This means we can use the press sound to divide the acceleration data and

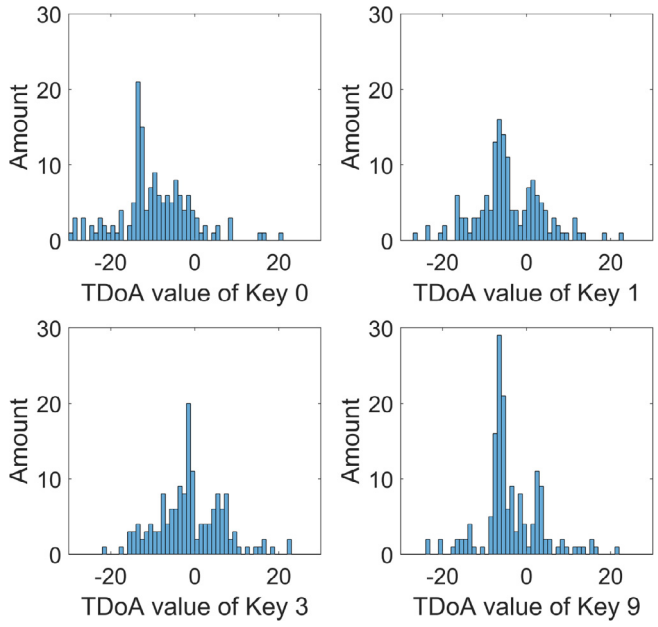

Fig. 7. TDoA value of Key '0', '1', '3' and '9'.

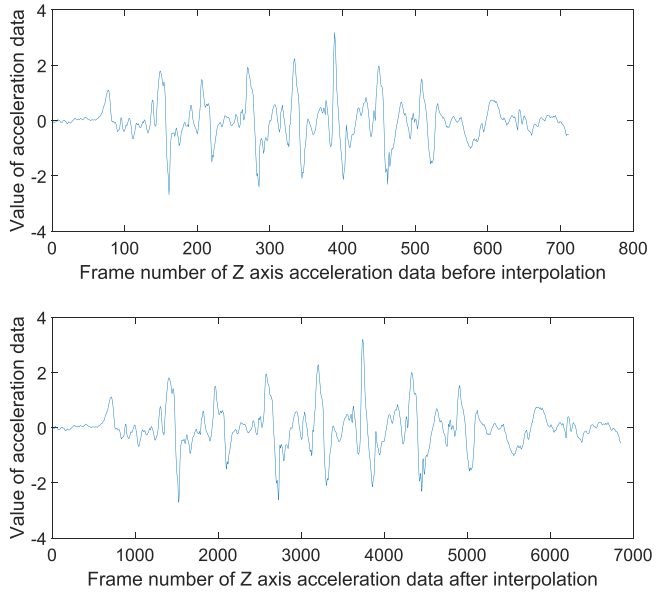

Fig. 8. Z axis acceleration data before and after interpolation.

know acceleration data between keys. In Section 3.1.2, we already detected the peak location of soundtracks recorded by the watch. We can use these peak locations to segment the acceleration data. Let L_n denote the location of the last peak value in acoustic data of smart wearable. A_{mn} denotes the data from position m to n in acceleration data, I_n denotes the location of the start of a movement, and MA_n denotes the nth movement. We can calculate I_n by:

$$I_n = L_n * (1/44100) * 1000 \tag{6}$$

After we get the start of the movements, we can segment those acceleration data MA_n:

$$MA_n = A_{I_n, I_{n+1}} \tag{7}$$

Fig. 9 shows the result of the above key segmentation method on the Z axis acceleration data. The red circles are the start and end index, e.g., the data between the first and second circle are the movement between the first keystroke and second keystroke,

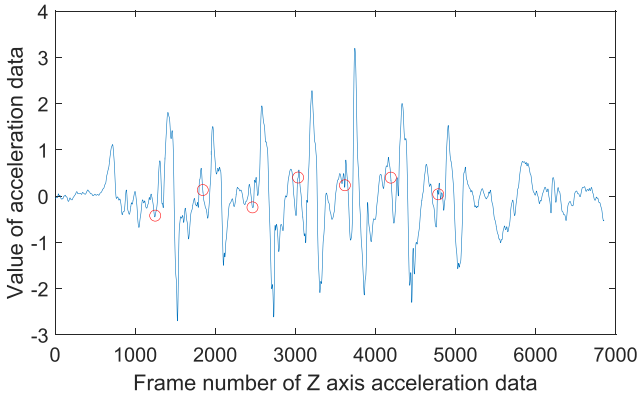

Fig. 9. Segmentation of Z axis acceleration data.

I_1 and I_2 of the above procedure. From Fig. 9 we can see that the circles are in the timing before we release the finger.

After the segmentation, we need to further process this data to make it feasible to be used in classifying the keystrokes. First we apply integration to this data to estimate the speed of the movements. Then we find the frame which has maximum absolute value in the speed data and select this frame as a feature. Let 'max()' denotes a function that returns the index of maximum value, $AF1_n$ denotes the nth movement's first acceleration feature:

$$i = max(\left| \int_{I_n}^{I_{n+1}} MA_n \, dn \right|) \tag{8}$$

Let t denote an array, t_k denotes k^{th} data in array t, we can get $AF1_n$ by:

$$t = \int_{I_n}^{I_{n+1}} MA_n \, dn \tag{9}$$

$$AF1_n = t_i \tag{10}$$

After that we apply double integration to MA_n, we mark the result as S_n:

$$S_n = \int_{I_n}^{I_{n+1}} \int_{I_n}^{I_{n+1}} M_n \, dndn \tag{11}$$

Let $AF2_n$ denote the nth movement's second acceleration feature after double integration:

$$AF2_n = S_{I_{n+1}} \tag{12}$$

We tested this double integration feature ($AF2_n$) using the 200-sample data set, which is mentioned in the TDoA test in Section 3.2.2. Fig. 10 shows the distribution of the X axis double integration features. From Fig. 10 we can infer the finger movement's direction by this data, e.g., movement '0' to '1' has a positive X value and movement '1' to '0' has a negative X value, which can be used in classifying keystrokes.

3.2.4. Angular velocity

Except for the accelerometer, the gyroscope is another movement sensor in the smart watch that is used to record the rotation changes. While moving a finger between keys, the rotation of the smart watch may change. Thus we suppose the angular velocity can also be used in movement recognition. The data format of the gyroscope is the same as that of the accelerometer, one frame including X, Y and Z axis data and a timestamp. First we interpolated the data to make it uniform. Then we used the peak value in the soundtrack to segment the data. Let L_n denote the location of the last peak value in the acoustic data of the smart wearable. G_{mn} denotes the data from position m to n in angular velocity data, I_n

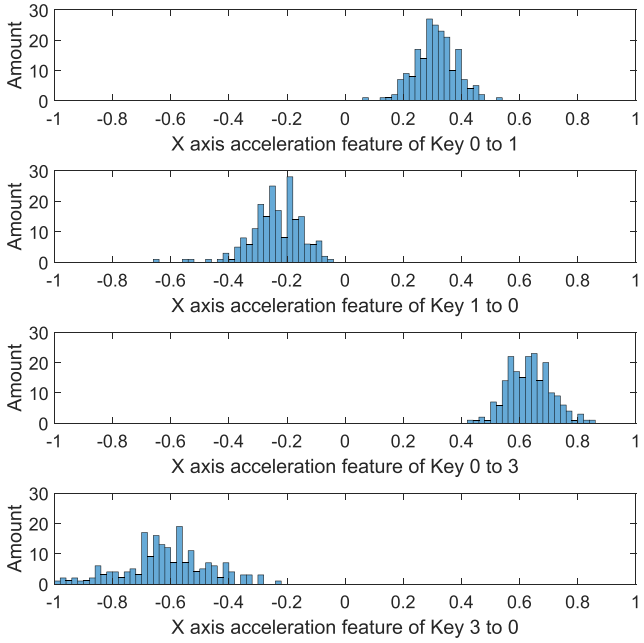

Fig. 10. Distribution of X axis acceleration features of '0103' PIN key.

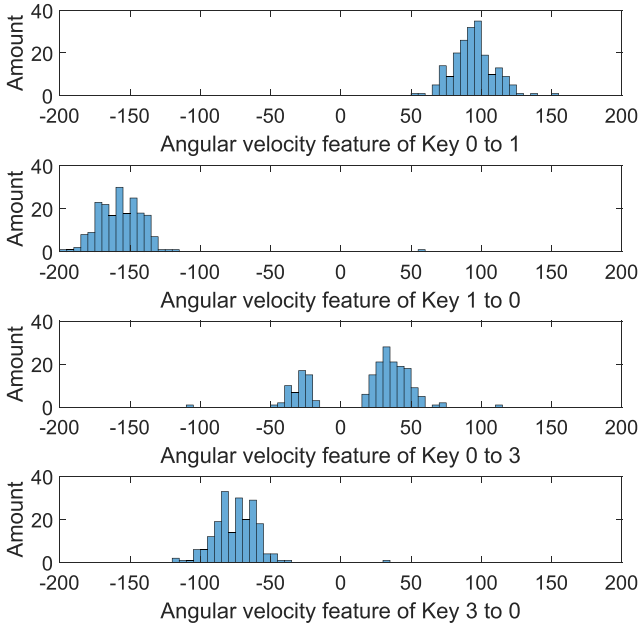

Fig. 11. Distribution of Z axis angular velocity features of '0103' PIN key.

denotes the location of the start of a movement and MG_n denotes the nth movement:

$$MG_n = G_{I_n, I_{n+1}} \tag{13}$$

After that we calculated the integral of the data and select the last frame in the movement as features, Let GF_n denote the nth movement's angular velocity feature, T denote an array, and T_k denote kth data in array T:

$$T = \int_{I_n}^{I_{n+1}} MG_n \, dn \tag{14}$$

$$GF_n = T_{I_{n+1}-1} \tag{15}$$

We used the 200-sample data set mentioned in Section 3.2.3 to test if this data can be used to distinguish movements. Fig. 11 shows the distribution of the double integration feature.

From Fig. 11 we infer some movements, e.g., most movements '0' to '3' have a positive Z axis value and movements '3' to '0' have a negative Z axis value. Besides, it seems that we cannot use X axis feature to distinguish keystrokes because we observed that the distribution is half positive and half negative.

After the feature extraction, we have 3 kinds of features: Features extracted from acoustic data, features extracted from linear acceleration and features extracted from angular velocity. The next section explains how to use these features to train classifiers and infer keystrokes.

3.3. Detail of key inference attack

After feature extraction from the raw data, We divided the data into two groups, a training set and a test set. The training set includes 15,000 single keystroke samples and the test set includes 3000 samples. Each sample contains these features:

1. TDoA of the mobile phone and smart wearable calculated using cross-correlation
2. TDoA of the mobile phone and smart wearable calculated using generalized cross-correlation
3. TDoA differences of the current key and the next key of the mobile phone and smart wearable calculated using cross-correlation
4. TDoA differences of the current key and the next key of the mobile phone and smart wearable calculated using generalized cross-correlation
5. X axis linear acceleration of movement from current key to next key
6. Y axis linear acceleration of movement from the current key to the next key
7. Y axis angular velocity of movement from the current key to the next key
8. Z axis angular velocity of movement from the current key to the next key
9. Next key

3.3.1. Attack procedures

As Liu et al. (2015) mentioned, when entering a PIN on an ATM keyboard, the last key that we enter must be the 'Enter' key, thus we can attack from the last key to the first key. First we infer the sixth key, the ninth feature of this sample is 'Enter' and we get an answer n. Next we classify the fifth key, the ninth feature is n. Repeat this procedure until the first key was classified. Fig. 12 shows the procedures of inferring a 6-digit PIN.

3.3.2. Machine learning model evaluation

In our experiment, we tried to use Decision Tree(Safavian and Landgrebe, 1991), K-Nearest Neighbor (KNN)(Keller et al., 1985), Neural Network(Abadi et al., 2016) and Support Vector Machine (SVM)(Noble, 2006) to classify the keystrokes. In Owusu et al. (2012), Owusu et al. showed that Decision Tree can be used in touch screen keystrokes inference. Similarly, Liu et al. (2015) used k-Nearest Neighbor to predict keystrokes on numeric keyboard, while Neural Network approaches has been demonstrated in the context of keystroke inference in both (Asonov and Agrawal, 2004) and (Xiao et al., 2020). SVM has been proved useful in classification problems(Dreiseitl and Ohno-Machado, 2002; Tang et al., 2008), so we also considered this approach as potential model in our keystroke inference case. We used the default configuration of the decision tree function 'fitctree' and the k-nearest neighbor function 'fitcknn' in MATLAB to train classifiers. Since these two function can return the label directly, we only need to train one model for both. The pseudo-

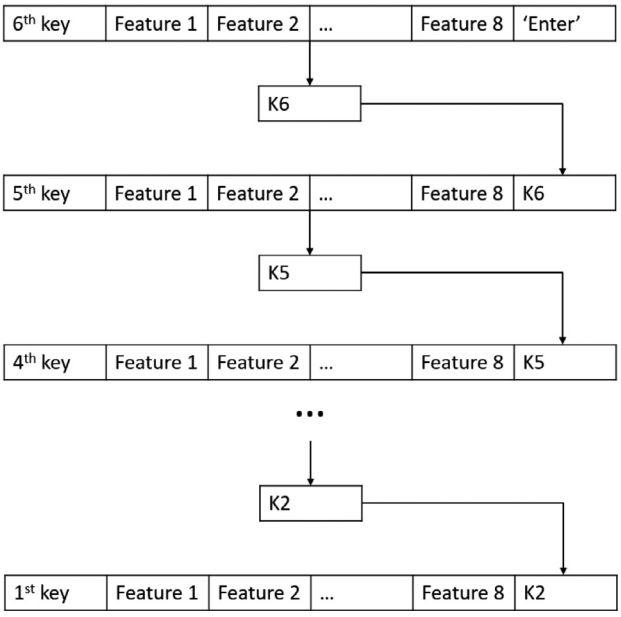

Fig. 12. Attack procedure inferring a 6-digit PIN.

codes for training and testing Decision Tree and KNN are showed in Algorithm 1 and 2 respectively.

Since there is no default configuration for the SVM classifier training function in MATLAB and Neural Network in Tensorflow(Abadi et al., 2016), we configured these two classifiers in the experiments. We used 'svmfit' API in MATLAB to train the SVM. The kernel function of SVM is a Gaussian kernel, the kernel scale is set to auto, which means we let the API decide the kernel scale. Since the API only support binary classification, we used the One vs. All strategy. We train 10 SVM classifiers, each classifier will evaluate if the input sample is a specified number and output a score. Each sample will be input to all 10 classifiers and get 10 scores, the classifier which has the highest score is the answer. The pseudo-code is showed in Algorithm 3

Algorithm 1 Pseudo-code for Decision Tree.

```
Input: x_train, y_train
  model = fitctree(x_train, y_train)
  i ← 0
  count ← 0
  while i < size(x_train) do
      label = predict(model, x_train(i))
      if label == y_train(i) then
          count = count + 1
      end if
      i = i + 1
  end while
  accuracy_Training_Set = count/size(x_train)
  (x_test, y_test) = load('test set')
  i ← 0
  count ← 0
  while i < size(x_test) do
      label = predict(model, x_test(i))
      if label == y_test(i) then
          count = count + 1
      end if
      i = i + 1
  end while
  accuracy_Test_Set = count/size(x_test)
```

Algorithm 2 Pseudo-code for KNN.

```
Input: x_train, y_train
  model = fitcknn(x_train, y_train)
  i ← 0
  count ← 0
  while i < size(x_train) do
      label = predict(model, x_train(i))
      if label == y_train(i) then
          count = count + 1
      end if
      i = i + 1
  end while
  accuracy_Training_Set = count/size(x_train)
  (x_test, y_test) = load('test set')
  i ← 0
  count ← 0
  while i < size(x_test) do
      label = predict(model, x_test(i))
      if label == y_test(i) then
          count = count + 1
      end if
      i = i + 1
  end while
  accurac_Test_Set = count/size(x_test)
```

Table 2
Accuracy of four classifiers.

Models	Training set	Test set
Decision Tree	94.15%	80.83%
KNN	100%	53.65%
SVM	89.18%	52.97%
Neural Network	98.50%	87.18%

We used Keras API from Tensorflow to train the Neural network. The neural network that we used has 4 hidden layers. The first hidden layer has 16 neurons, the second hidden layer has 64 neurons, the third has 256 and the forth has 64. All hidden layers has a kernel regularizer with parameter 0.0001. The activation of all hidden layers are 'relu'. The pseudo-code of Neural Network is showed in Algorithm 4.

We used the 18,000 keystrokes collected by one person to test these models, 15,000 keystrokes used as training set, the other 3000 keystrokes used as test set. This is done as we are only interested at this stage in comparing the effectiveness of models and features, and not yet adversarial key recovery (where the users in training set is different to testing set). We used all features to train these four classifiers. Decision Tree has 94.15% accuracy in the training set and 80.83% in the test set. KNN has the highest accuracy in the training set, but performed worse than Decision Tree in the test set. SVM has the same problem of KNN, high accuracy in the training set but low accuracy in the test set. The Neural Network has 98.50% accuracy in the training set and 87.18% in the test set. The result is showed in Table 2.

From the result we can see that the Neural Network has the best overall performance. It will output a vector of marks, the position of the highest mark is the predicted answer. Its accuracy in the training set is the second highest value and its accuracy in the test set is the highest one. Hence we decided to use the Neural network in our further experiments.

3.3.3. Feature evaluation

In the last paragraph we showed the accuracy of using all features to classify keystrokes. However, in the real world sometimes we cannot collect all the data, e.g., the ambient noise is too large

Algorithm 3 Pseudo-code for SVM.

```
Input: x_train, y_train
  SVMModel[10]
  i ← 0
  while i < 10 do
      SVMModel[i] = fitcsvm(x_train, y_train(:, i),
      'Standardize',true,'KernelFunction',
      'RBF','KernelScale','auto');
      i = i + 1
  end while
  i ← 0
  count ← 0
  while i < size(x_train) do
      j ← 0
      label ← 0
      score ← 0
      while j < 10 do
          tempScore = predict(SVMModel[j], x_train)
          if tempScore >score then
              score = tempScore
              label = j
          end if
      end while
      if label == y_train(i) then
          count = count + 1
      end if
  end while
  accuracy_Training_Set = count/size(x_train)
  (x_test, y_test) = load('test set')
  i ← 0
  count ← 0
  while i <size(x_test) do
      j ← 0
      label ← 0
      score ← 0
      while j < 10 do
          tempScore = predict(SVMModel[j], x_test)
          if tempScore >score then
              score = tempScore
              label = j
          end if
      end while
      if label==y_test(i) then
          count = count + 1
      end if
  end while
  accuracy_Test_Set = count/size(x_test)
```

Algorithm 4 Pseudo-code for Neural Network.

```
Input: x_train, y_train
  model = tf.keras.models.Sequential()
  model.compile()
  model.fit(x_train, y_train)
  accuracy_Train = model.evaluate(x_train, y_train)
  (x_test, y_test) = load('test set')
  i ← 0
  count ← 0
  while i <size(x_test) do
      j ← 0
      label ← 0
      scores[10] = model.predict(x_test)
      score ← 0
      while j < 10 do
          if scores[j] >score then
              score = scores[j]
              label = j
          end if
      end while
      if label==y_test(i) then
          count = count + 1
      end if
  end while
  accuracy_Test_Set = count/size(x_test)
```

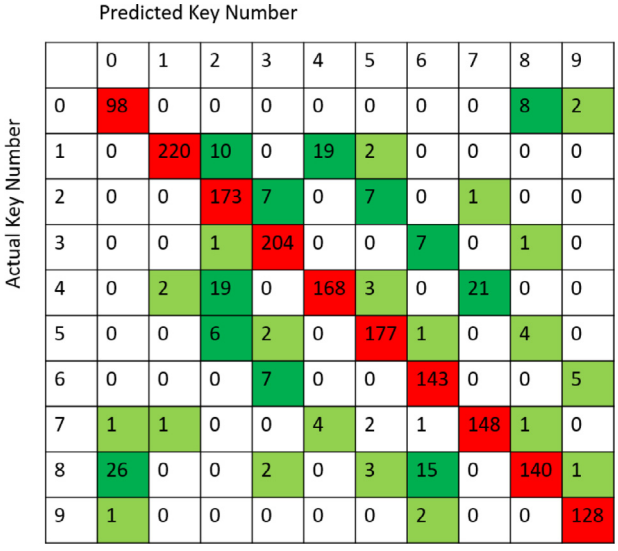

Fig. 13. Statistic of Single Keystroke Inference Result.

Table 3
Accuracy of using different feature groups to predict keystrokes .

TDoA	Linear Acceleration	Angular velocity	Training Set	Test Set
√	√	√	98.50%	87.18%
√	√	×	78.66%	51.88%
√	×	√	80.82%	67.65%
×	√	√	94.11%	88.16%
√	×	×	58.97%	26.50%
×	√	×	60.65%	48.70%
×	×	√	68.86%	66.08%

and interfere with the acoustic data collection, or due to the hardware limitation we cannot collect angular velocity. Thus we tried to train the neural network classifiers using a single feature and two features. Table 3 shows the result of predicting a single digit using different feature groups, in the first 3 columns, '√' means this feature is used, '×' means feature not used, the forth column shows the accuracy.

From Table 3 we can see that if we use all features to train the classifier and predict the test set, the accuracy is 87.18% on predicting a single digit. Fig. 13 shows a statistic of this single keystroke inference result. The column number means the predicted key number and the row number means the actual key number, e.g., the number row '8' column '0' means 26 keystroke '8' are predicted to be key '0'. When using TDoA and linear acceleration, the accuracy is 51.88%, less than 67.65% accuracy of using TDoA and angular velocity. If we use a single feature, the accuracy of TDoA, linear acceleration and angular velocity are 26.50%, 48.70% and 66.08% respectively. An interesting thing is, when we use linear acceleration and angular velocity, the accuracy can reach 88.16%. In the next section, we therefore use a combination of linear acceleration and angular velocity to do adversarial inference.

Table 4
Accuracy of Predicting PINs Entered by Different Users.

User	1	2	3
Single Digit	73.3%	76.7%	74.3%
6-Digit PIN	50%	60%	46%

3.3.4. Adversarial recovery results

This section presents the PIN entry results from the method we considered the most promising following initial evaluation in the previous section. This method primarily uses linear acceleration and angular velocity to recover keystrokes entered with the acoustic data only used for keystroke segmentation (not TDOA). In practice, the required sensor data can therefore be obtained from a single smart watch (microphone, accelerometer and gyroscope) equipped by the user entering the PIN. Alternatively the sound data could be from a mobile, also on the user's person, and combined with the movement data from the smartwatch. So far we only evaluated the comparative ability of machine learning methods and feature groups to do key inference using a training set and testing set by the same user. In this section, we evaluate the ability to do adversarial key inference where the attacker can train the model on its own data and then attempt to do keystroke recovery of other users in the testing set. We assume a basic adversarial model where the attacker has compromised the user's wearable and mobile devices to collect movement and audio data during the key entry. The attacker does not have access to the actual key entry device to generate the training set, but could have a similar device, e.g. same model/size.

For the inference experiment we use a training set of 18,000 keystrokes, as described in Section 3.3.2. The testing set comprises of keystrokes gathered from three users, two of which entered 10 random 6-digit PINs and a third who entered 50 random 6-digit PINs. The third user with more PINs is to evaluate whether the number of PINs entered significantly changes the inference accuracy. We predict single keystroke entries we used the method in Section 3.3.1 to infer the entire 6-digit PIN keys in the test set. We use a neural network with linear acceleration and angular velocity features as we discussed in Sections 3.3.2 and 3.3.3, while using audio data to detect and segment the key stroke events. In the real world when entering PINs the attacker most often has 3 chances, so we follow the method in (Liu et al., 2015) to consider the inference a success if the PIN can be correctly inferred after three guesses. For single keystroke inference the accuracy reflects the number of times the keystroke was inferred on the first guess. The results for single keystroke and 6-digit PIN recovery for the three users are shown in Table 4.

Since the training set mentioned above is generated by one user (assuming the situation where the attacker generates a training set alone), there is a possibility that the model has a bias towards specific movements of the training set user, which could negatively effect the accuracy of predicting other users' keystrokes in the testing set. To investigate this issue, we did a basic proof-of-concept experiment. Five users were invited to enter a 6-digit PIN 20 times each. We then had five trials, each time training the model with the 120 keystrokes (6x20) from one user and then inferring the single keystrokes entered by the other four users. The average accuracy of the five trail runs was 85.6%.

We then again set up five trials, in each case using the keystrokes from four users to train the model and then using it to infer the single keystrokes collected from the remaining user. The average accuracy of the five trail runs was 95.3%. We potentially see that involving more users in training can improve the accuracy, possibly because training on more users tends to produce a more "average" model of user movements rather than potentially being biased towards specific movement habits of a single user. It should be noted that the accuracy of this experiment is not comparable to that in Table 4. That was inference on keystrokes from any random 6-digit PIN, which might not even be in the training set, whereas here we are using a single 6-digit PIN sequence. So it expected that the accuracy will increase, so the absolute value does not matter but the interesting outcome is that this value increased with more users in the training set.

3.3.5. Limitations

Key inference using sensor data does have limitations in making it a widespread attack method. These limitations apply to our method and also related works using the same features. Collecting sensor data from a smartwatch requires an application in the watch and on the phone. In newer devices this requires the user to install a separate application for each device, although it is feasible that an attacker could work around this, e.g. what appears on the surface to be a fitness application could prompt the user to install both the mobile and watch components. For the key inference to be made the user must wear the watch on the hand entering the PIN, which might not be case as many people prefer to wear their watch on the non-dominant hand, and enter the PIN by making a hand movement instead of only using finger movement. If sound features are used the keypress action should make a noise, either from the mechanical movement of the key or the finger hitting the key. The latter sound is often present even if the keys are soft or the keyboard is considered "silent". We considered TDOA as a possible feature, although in our method we did not end up using it. If TDOA was to be used the mobile phone (the second reference point) should remain still while the PIN is entered.

4. Discussion of related work

Our work is related to context-free keystroke recovery so we confine our discussion to work in this area. In the context-free case the attacker does not have any further information about the keystrokes entered other than what is learned from the observed sensor data, e.g. if we have a 6-digit PIN the possibility of each PIN being entered is equally likely. Doing key recovery with context uses additional information about the keystrokes being entered to improve the attacker's recovery, e.g. the user is typing an English passage so we could use letter frequency statistics and dictionary words to aid in keystroke guessing. In our comparison, we also focus on works where the attacker does not need to have physical access to the actual keypad device to be attacked, i.e. the attacker trains on a similar keypad. Examples of such works are Asonov and Agrawal (2004),who trained a neural network on the unique sound features of a specific key (each key pressed would make a different sound) and later recovered key presses on the same keyboard, and de Souza Faria and Kim (2019) who used features of audio signals from two microphones installed within the device to train a Gaussian Naïve Bayes classifier for key press recovery. However, we still include Owusu et al. (2012), as even though it used a sensor inside the device it is feasible that this data could be remotely compromised.

Zhu et al. proposed a context-free keystroke inference method (Zhu et al., 2014). Different from previous mentioned methods, they used the time differences of arrival to locate the sound source. They record the sounds by two or more phones in different locations and then find the hit peaks in each soundtrack. They used generalized cross correlation to decide the arrival time of the keystroke sound. After the TDoA features were calculated, they estimate the distances between keys and microphones, then tried to estimate the keys position on the keyboard by calculating the intersection of distances from the key to the microphones. The re-

Table 5
Comparison between related works.

Paper	Sensor Used	Input Type	ML Model	Accuracy	Remark
(Zhu et al., 2014)	Microphone	Key press	None	72% on single keystroke	Need to know location of two microphones
(Liu et al., 2015)	Accelerometer	Key press	k-Nearest-Neighbor	45%-65% on 6-digit keystroke	Predicting 6-digit keystroke in 3 guesses Special training sequence
(Owusu et al., 2012)	Accelerometer	Touch screen	Random Forest	24.5% on touch areas	Predicting touch areas Accelerometer inside device
(Xiao et al., 2020)	Microphone	Touch screen	Neural Network	76.4% on single keystroke	Needs 192kHz sampling rate microphone
(Wang et al., 2016)	Accelerometer	Key press	None	80%* on 4-digit keystroke	Need accurate sampling rate (not guaranteed in current devices/API) Train on specific keyboard Predicting 4-digit keystrok in 3 guesses (*device within "normal" rate restriction 50 Hz Google Sensor API)
(Liu and Li, 2019)	Accelerometer	Key press	Deep Neural Network	70% on set of short PINs (4–6 digits)	Prediction in 3 guesses Sampling less than 50 Hz 6-digit with known keyboard size/button amount ≈ 43 minutes, can optimise search as two 3-digit with reduced accuracy
Our method	Microphone, gyroscope, accelerometer	Key press	Neural Network	74% on single keystroke 46–60% on 6-digit keystroke	Normal mobile microphone (44.1 kHz sampling) Predicting 6-digit keystroke in 3 guesses

sult showed that this method can correctly infer 72% of keystrokes. From their work we can see it is possible to use TDoA to classify keystrokes. In comparison, our work has slightly higher keystroke success without the need to know exactly where the recording devices are (in our work we roughly know where the devices are located relative to the keypad).

Liu et al. suggested a side channel attack on a numeric keypad by using an accelerometer on a smart watch(Liu et al., 2015). They recorded 41 vector movements between 10 number keys and the 'Enter' key, then re-sample the non-uniform acceleration data by interpolation. After that they applied FFT to these data to remove noise and random movements of human hands. They recorded a total of 4920 movements, including 3720 motions between two numbers and 1200 between numbers and the "Enter" button. They chose the k-Nearest-Neighbor to classify movements, with selected k from 5 to 50. With the feature input, the classifier would output all possible movements and a probability, which also took into account the likely previous movement. Their result showed that the success rate range of guessing the PIN in 3 tries for three users is between 45% and 65%. Their work shows that using an accelerometer in smartwatch to attack PEDs is a practical method. We achieve similar success in key recovery but we only require a training set consisting of a number of random PINs entered rather than specific movement vectors. This arguably simplifies covertly building a larger training set, e.g. it is easier to trick people into entering number sequence while recording movement and/or sound than repetitively making vector movements.

Owusu et al. suggested accelerometers can be used to infer keystrokes on virtual keyboards(Owusu et al., 2012). They record the acceleration data with timestamps while a user is touching the buttons. The key touch movements were segmented by the timestamps when a button was touched. They used the acceleration data during a keypress as a single sample to do the classification. Their result showed that the Random Forest classification algorithm performed best among the tested classifiers, including a multilayer perceptron artificial neural network, sequential minimal optimization trained Support Vector Machine and C4.5 decision tree. The accuracy for predicting a single touched area is 24.5%. They also investigated the effect of sampling rate. In their observation, the accuracy improved significantly when the sampling rate increased from 50Hz to 100Hz. They showed that a decision tree can be used in key inference. Our work also leverages an accelerometer but it measures user hand movements, not the acceleration of the key entry device, and yields more accurate results.

Xiao et al. introduced a method of inferring keystrokes on the virtual keyboard of a mobile phone's screen(Xiao et al., 2020). They used the TDoA and spectrum to guess the keystroke area on the screen. Different from our work, the microphone that they used had a much higher sampling rate of 192 kHz, which will make the TDoA estimation more accurate. They used an unsupervised neural network to classify the keystrokes. Compare to their work, our method is more practical, since the 192kHz sampling rate is not a popular standard in a mobile phone's microphone, whereas our method used a standard 44.1 kHz microphone.

Wang et al. introduced a method that can recover PIN entries using wearable devices based on distance estimates from acceleration data(Wang et al., 2016). First, they use the press and release movement of the keypress to segment the movement data. Then they apply double integration to the acceleration data to estimate the distance. This is used in sub-path calculation and PIN recovery using a trajectory inference tree. This offers excellent results of 80–94% accuracy considering top three guesses for a 4-digit PIN across three devices tested (25 Hz, 100 Hz and 200 Hz sampling rate). However, this method is reliant on accurate and fixed sampling rate and to obtain best results require higher sampling rates. This is no longer guaranteed with sensor data being limited due to security concerns, e.g. Android 12 rate limits accelerometer data with normal rate being 50Hz (one test device in this paper meets this constraint), and higher rates requiring further explicit security permissions. It also currently states that any fixed sampling rate specified is considered only as a "hint to the system" and actual sampling rate would vary between 50% and 220% of nominal value. The method also requires data collected on the targeted keypad of the same size, while other methods, including ours, just need keypads with the same key layout regardless of size.

Liu et al. used the accelerometer in the smart watch to recover typed keystrokes(Liu and Li, 2019). In their observation, the typing style of a password is generally fast and smooth, hence they used the interval threshold to determine the click peaks in the acceleration data and used these peaks to segment data. Next, they

further used the movement data to train a deep neural network to infer keystrokes achieving accuracy of approximately 70% for 4-6-digit keystrokes. This work introduces some interesting aspects, such as considering the effects of keypad size, searching longer PINs by combining shorter segments, and offering the option of not needing a known final key pressed. This comes at a cost of exponential computational time, with a 6-digit PIN inference using the conventional approach to the proposed method requiring about 43 minutes of not split up into a search of smaller sequences, which could lead to increased error, i.e. error for 6 digits e_6 as two 3-digit searches is $(e_3)^2$.

From the above related works, we can see that acoustic and acceleration data can be used to infer keystrokes. However, these works are using only single sensor sources (acoustic or acceleration). Our work (See Section 3.3.3.) investigated the potential benefits of using a fusion of 3 kinds of sensors, including sound, linear acceleration and angular velocity. Compared to these related works, our method has a 74% accuracy on predicting single keystroke and 46% to 60% on predicting 6-digit PINs. Table 5 shows the comparison between related work and our paper.

5. Conclusion

In this paper, we introduced an attack method for predicting context-free random short numbers entered on a keypad using sensor data from the microphone, accelerometer and gyroscope in the mobile phone and smart watch of the user entering the number. As with related works, it is assumed that malicious software, e.g. mobile malware or a malicious applications, could collect and synchronise the sensor data from the two devices. Firstly, we evaluated potential attack methods based on features from sensor data on mobile and wearable devices. We used extracted features to train four classifiers including the Decision Tree, the K-Nearest Neighbor, the Support Vector Machine and the Neural Network and found that the Neural Network had the best overall performance. After that, we investigated the impact of combining different features and found the best result was obtained from linear acceleration and angular velocity of the entering hand together with key press segmentation (identifying each individual key entry period) with acoustic data. Finally, we tested this attack method on different users and we successfully recover 74% single keystrokes in one prediction and 46%-60% of 6-digit PINs in 3 predictions. We also did a supplementary experiments which showed that involving multiple users in training set can improve the accuracy. Our result in recovering single keystrokes and keystroke sequences (for 4 and 6 digit PIN entry) is an improvement on existing work in literature. This is the first method to use angular velocity in keystrokes inference and also demonstrates the potential of using the fusion of different features to guess keystrokes. In the future, we will work on security implementations based on sensors in smart devices. A possible direction is key derivation using an accelerometer and a gyroscope, since the human body may move randomly and generate random factors in the data, which can be used in random number generation.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Yuanzhen Liu: Conceptualization, Methodology, Software, Validation, Investigation, Writing – original draft. **Umair Mujtaba Qureshi:** Conceptualization, Writing – review & editing. **Gerhard Petrus Hancke:** Conceptualization, Writing – review & editing, Supervision.

Acknowledgement

This work was supported by City University of Hong Kong Project CityU 11218419 (7005219) in collaboration with the funded-research project SEHS-2021-234(E) from the College of Professional and Continuing Engineering (CPCE), an affiliate of The Hong Kong Polytechnic University.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D.G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., Zheng, X., 2016. Tensorflow: a system for large-scale machine learning. In: 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16). USENIX Association, Savannah, GA, pp. 265–283.

Alpaydin, E., 2020. Introduction to Machine Learning. MIT press.

Asonov, D., Agrawal, R., 2004. Keyboard acoustic emanations. In: IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004. IEEE, pp. 3–11.

Azaria, M., Hertz, D., 1984. Time delay estimation by generalized cross correlation methods. IEEE Trans. Acoust. 32 (2), 280–285.

Bond, M., Choudary, O., Murdoch, S.J., Skorobogatov, S., Anderson, R., 2014. Chip and skim: cloning EMV cards with the pre-play attack. In: 2014 IEEE Symposium on Security and Privacy. IEEE, pp. 49–64.

Dreiseitl, S., Ohno-Machado, L., 2002. Logistic regression and artificial neural network classification models: a methodology review. J. Biomed. Inform. 35 (5–6), 352–359.

Drimer, S., Murdoch, S.J., Anderson, R., 2008. Thinking inside the box: system-level failures of tamper proofing. In: 2008 IEEE Symposium on Security and Privacy (sp 2008). IEEE, pp. 281–295.

Joy Persial, G., Prabhu, M., Shanmugalakshmi, R., 2011. Side channel attack-survey. Int. J. Adv. Sci. Res. Rev. 1 (4), 54–57.

Keller, J.M., Gray, M.R., Givens, J.A., 1985. A fuzzy k-nearest neighbor algorithm. IEEE Trans. Syst. Man Cybern. (4) 580–585.

Liu, X., Zhou, Z., Diao, W., Li, Z., Zhang, K., 2015. When good becomes evil: keystroke inference with smartwatch. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pp. 1273–1285.

Liu, Y., Li, Z., 2019. Aleak: context-free side-channel from your smart watch leaks your typing privacy. IEEE Trans. Mob. Comput. 19 (8), 1775–1788.

LIU, Y., QURESHI, U.M., Hancke, G.P., 2021. Feasibility of inferring keystrokes on peds with sensors from mobile devices. The 30th IEEE International Symposium on Industrial Electronics (ISIE). IEEE.

Murdoch, S.J., Drimer, S., Anderson, R., Bond, M., 2010. Chip and pin is broken. In: 2010 IEEE Symposium on Security and Privacy. IEEE, pp. 433–446.

Noble, W.S., 2006. What is a support vector machine? Nat. Biotechnol. 24 (12), 1565–1567.

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, pp. 1–6.

Panda, S., Liu, Y., Hancke, G.P., Qureshi, U.M., 2020. Behavioral acoustic emanations: attack and verification of pin entry using keypress sounds. Sensors 20 (11). doi:10.3390/s20113015.

PCI, P. C. I., 2011. PIN Transaction Security (PTS) Point of Interaction (POI).

Qureshi, U.M., Hancke, G.P., Gebremichael, T., Jennehag, U., Forsstrm, S., Gidlund, M., 2018. Survey of proximity based authentication mechanisms for the industrial internet of things. In: IECON 2018 – 44th Annual Conference of the IEEE Industrial Electronics Society, pp. 5246–5251. doi:10.1109/IECON.2018.8591118.

Safavian, S.R., Landgrebe, D., 1991. A survey of decision tree classifier methodology. IEEE Trans. Syst. Man Cybern. 21 (3), 660–674.

de Souza Faria, G., Kim, H.Y., 2019. Differential audio analysis: a new side-channel attack on pin pads. Int. J. Inf. Secur. 18 (1), 73–84.

Tang, Y., Zhang, Y.-Q., Chawla, N.V., Krasser, S., 2008. Svms modeling for highly imbalanced classification. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 39 (1), 281–288.

Teh, P.S., Zhang, N., Teoh, A.B.J., Chen, K., 2016. A survey on touch dynamics authentication in mobile devices. Comput. Secur. 59, 210–235.

Wang, C., Guo, X., Wang, Y., Chen, Y., Liu, B., 2016. Friend or foe? Your wearable devices reveal your personal pin. In: Proceedings of the 11th ACM on Asia conference on computer and communications security, pp. 189–200.

Xiao, Z., Chen, T., Liu, Y., Li, Z., 2020. Mobile phones know your keystrokes through the sounds from fingers tapping on the screen. In: 2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS). IEEE, pp. 965–975.

Zhu, T., Ma, Q., Zhang, S., Liu, Y., 2014. Context-free attacks using keyboard acoustic emanations. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, pp. 453–464.

Zhuang, L., Zhou, F., Tygar, J.D., 2009. Keyboard acoustic emanations revisited. ACM Trans. Inf. Syst. Secur. 13 (1), 1–26.

Mr. Liu Yuanzhen I am currently a PhD student in the Department of Computer Science at City University of Hong Kong. I obtained degree of Bachelor of Management from Guangdong University of Foreign Studies. I obtained the degree of Master of Science from City University of Hong Kong. I am interested in Information Security of Smart devices and Internet of Things.

Dr. Umair Mujtaba Qureshiis a Lecturer at the Division of Science, Engineering and Health Studies at The Hong Kong Polytechnic University. Dr. Qureshi obtained his PhD degree in Computer Science from the City University of Hong Kong in 2020, MEng degree in Communication Systems and Networks and BEng degree in Telecommunication Engineering from the Mehran University of Engineering and Technology, Pakistan in 2014 and 2010 respectively. His research interests include Applied Data Science and Analytics, Internet of Things, Wireless Communication, and Sensor Networks and Security.

Dr. Gerhard Petrus HANCKEI am currently an Associate Professor in the Department of Computer Science at City University of Hong Kong. I obtained B.Eng and M.Eng degrees from the University of Pretoria (South Africa) in 2002 and 2003 respectively, and a PhD in Computer Science with the Security Group at the University of Cambridge's Computer Laboratory in 2009. Before joining CityU I worked for the Information Security Group at Royal Holloway, University of London as a researcher/engineer at the ISG Smart Card Centre (2007–2011) and as a Teaching Fellow (2011–2013).