## Chapter 3

## 3.1 Android password related mechanism

Android uses the following authentication keys to ensure password security

**3.1.1 Key storage**

Android stores key (cryptographic) and ensures standard routines. Android hardware support and backup the storage of keys. Due to the hardware support it is difficult to find android keys. As these keys are directly related to password and it’s security, it is difficult to crack android lock screen password. Android not only provide hardware backed storage of keys but also provides keymaster for security services. It provides execution environment which is trusted and secure services ( example strongbox).

**3.1.2 User authentication**

Android provides pin or pattern, biometric as well as face unlock. Android keeps gate keeper for pin and pattern authentication and finger print for biometric authentication. In android devices which are using above nougat and above can use biometric prompt for both fingerprint and biometric data. The authentication states of these components are communicated with the key store service.

**3.1.3 Credentials enrollment**

Whenever user boots device for the first time, i.e. after factory reset all authenticators like key, biometric or face are ready to receive credentials enrollment from user. A user is required to provide these details with authenticators. User has to provide atleast pin or pattern details, other details may be shared after some time or according to user preferences. This created a secure random identifier of 64 bit which is known as SID ( secure identifier). This work as a token which binds user and it’s cryptography. SID binds to user password cryptographically. Whenever a user authenticates successfully with Gatekeeper, Auth tokens are generated. These contain SID of the user and help in successful authentication.

If user wants to change the credentials then he has to provide previous credentials. After successfully verifying the previous credentials the SID attached with old credentials is transferred to the new credentials.

The security of android is of very high level, so directly cracking pin is not so feasible. One way of exploiting android pins is through 0 permission sensors. These are the sensors which don’t notify user about their usage, and can be accessed by any app or website (through browser) silently.

We can analyse data of these sensors to find out which part of the screen is pressed. The location of screen can be used to find out the particular key, and help in cracking the key.

## 3.2 0-permission sensors

Today we have various operating systems on our phones which includes the popular Android and iOS. According to the survey conducted by StatCounter, in March 2020, there are 72.26% Android users, 27.03% iOS users and 0.71% other mobile operating systems. We will be using Android phones that also shows that we will be targeting the larger group of devices.

By creating an Android app (GyroAccData) for the phone and using OnePlus 5T and Redmi Note 5 Pro in every experiment of data collection. Learning from the Android Developers documentation [14] about the motion and position sensors and implementing it in the app, enabling the sensors, namely gyroscope, gravity and accelerometer sensor on the foreground of the app and display a live graph of the data being generated by them. Whenever a SensorListener is registered, and an event is generated when there is motion detected for sensors. It is handled by onSensorEvent.

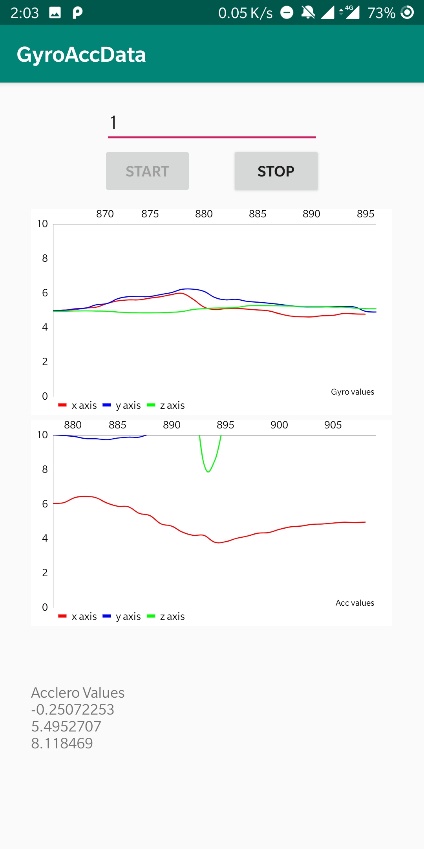


Figure 1: Sudden rise is seen in the graph when the screen is touched.

The top graph shows the gyroscope, and the bottom graph shows accelerometer events.

All the sensors were kept at maximum frequency, and it was observed from the graphs that on a touch of the screen it creates a spike in the graph as it generates small movements when the screen is touched and released.

In the process of creating this app and checking the functions of the sensors, there was no single permission asked or any message shown on the screen that tells a user that these sensors are being used. Many such sensors are being used for activity tracking like step counting, augmented reality which also requires great computation power. Still, with advancements in processing power, the security measures are set aside for a while and not paid attention to with this detail.

These sensors are called zero-permission sensors which can leak sensitive information and, in our case, we can get the Personal Identification Number (PIN) from the user and his/her motion patterns.









Figure 2: Level of self-declared knowledge about different mobile sensors

Figure 5., a survey conducted and published in [15] (Fig.5), we can see that majority of the people have never even heard about these sensors and hence the vulnerability increases.

## 3.2 Using sensors for predicting touch

From Figure 5, we know that it is possible to see when the screen is being touched. Using the high-frequency sensors, it is also possible to know what portion of the screen is being touched by the user by looking at the specific pattern in the motion that is being recorded by these sensors. Using the same app (GyroAccData) with modifications that store the data generated by the sensor events and storing them in (.csv) file format, we observe the following graph.

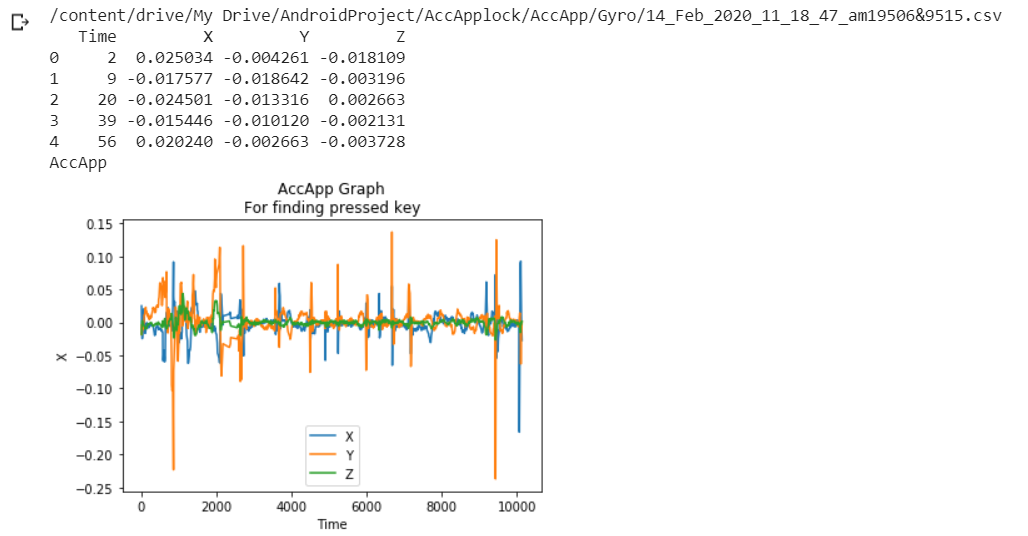


Figure 1: Unique spikes in the graph when different keys are pressed

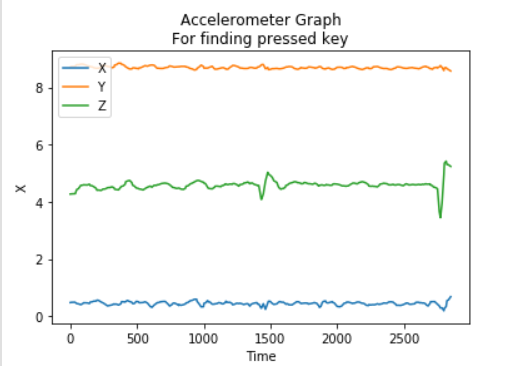
It was observed that when the phone is held in one hand and kept stable and when numeric keypad (on-screen keyboard) keys are pressed, a unique sensor data pattern is produced which can be used to predict which part of the screen is being touched.

This experiment was tried in different environments and with multiple users. As the number of users increases, the way the phone is held changes. The person can be left-handed, right-handed, may keep the phone in both the hands, use it in one hand, touch with both thumbs or just one or use index finger. By keeping the phone on a stationary material like a table, it was observed that due to the bulge of the camera module at the rear of the phone and also due to the protective cover (may or may not be present), the spikes in the graph were observed, suggesting there might be a chance of predicting data even if it is kept on some stationary material. Different profiling can be done to the way the phone is being held, and machine learning classification models can power the prediction.

After observing the graphs, we can say that whenever a key is pressed the sufficient spike is observed in accelerometer, gyroscope and gravitometer graph. The size of spike differs according to the key pressed and the orientation of the phone. As sensors collect different values of x, y, and z axis whenever different keys are pressed and if the orientation of phone is same in the hand then the values for same keys are similar.

Using this approach, we can find that whenever similar keys pressed sensors record same amount of variation in x, y and z axis. Using this information, we can predict the key pressed, when these sensors data are passed into a machine learning model.

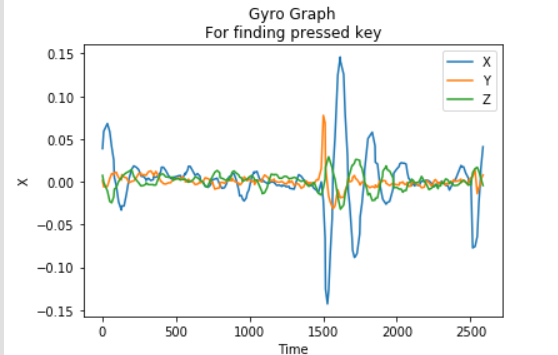
Here I am showing variations in sensors data when a particular key is pressed.



*Figure 2: Unique spikes in the graph when a key is pressed in accelerometer sensor.*

As we can see that there is sufficient spike observed in graph whenever a key is pressed. The z axis of accelerometer shows maximum deflection, as there is significant movement in z direction.

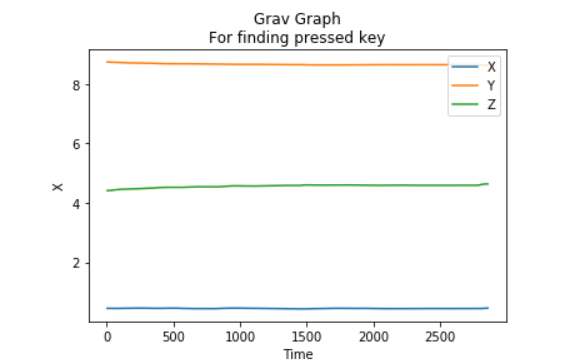
x and y axis also show a change but it is less as compared to z direction. The changes which are showing at the initial and final end are due to the movement of phone while picking up and taking down.



*Figure 2: Unique spikes in the graph when a key is pressed in gyroscope sensor.*

As seen in the above graph we can see that whenever a key is pressed, a sufficient deflection is shown in gyroscope graph. The largest deflection is shown in x axis of the gyroscope component.

However y and z direction also shows some deflection.



*Figure 3: Unique spikes in the graph when a key is pressed in gravitometer sensor.*

As seen in the above graph, whenever a key is pressed there is very less deflection observed in gravitometer sensor. However, the deflection is not zero. We can combine gravitometer data with other sensors to get good accuracies.

**3.3 Ways to exploit leakage**

Exploring how to exploit a leakage like this to steal PINs. There are two possible approaches. Either we can make a website that uses sensors in the background, or we can build an Android application.

**3.3.1 Site**

As seen in the implementation of the paper [15], they implemented a web page with embedded JavaScript code to collect the data from voluntary users. On the client-side, they had developed a GUI in HTML5, which brings up the virtual keyboard for users and then randomly generated 4-digit PINs are to be entered by the users. [15]

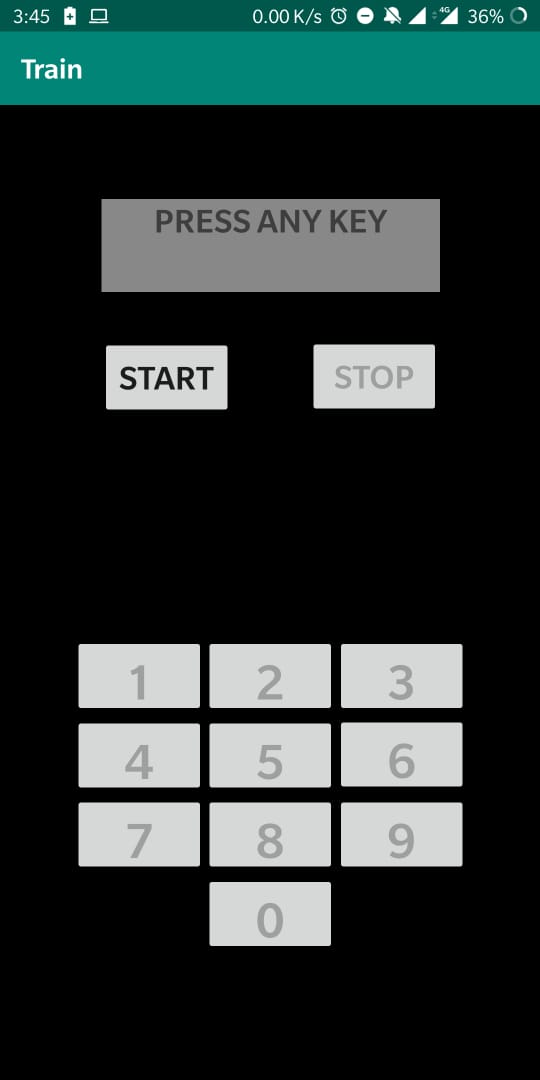


*Figure 4: Site opened on a phone*

The sensor data is sent to the database along with the PIN for which the data is held by the file. In a real-world attack scenario, we need to identify the keypress timestamp from the peak amplitudes in the data as the actual timestamp will not be recorded. We can see such implementation in [16].

**3.3.2 Android Application**

An android application “Train” created by us, uses the Gyroscope, Accelerometer and Gravity sensors to collect data from the users. As soon as the person press the “start” button, the sensor listeners are registered and start saving the data generated in (.csv) file format, with each file containing data for one key press only. As seen in Figure 8, the app also provides a numerical keypad for the user to enter PINs.



*Figure 5: “Train” application that takes sensor data in background*

In the training phase, a user is made to hold the phone in one hand and press the keys as displayed on the phone. Each key is to be pressed four times once the “start” button is pushed and all the ten digits having four files each is stored on the external storage. A user press “stop” button and all sensors are unregistered and stop data collection. After each keypress, the corresponding file is saved with the label key press, key down and key up timestamp. Using the onKeyDown and onKeyUp events under Android activity, we can get the required timestamps.

In the future implementation of an actual attack, we can deploy a service that will be running in the background that makes use of sensors and collect data from the user using similar approach from the section 3.3.1. It is discussed in details further in the report.

*Figure 2: Unique spikes in the graph when a key is pressed in gravitometer sensor.*