Android Pin Cracking

Project report submitted in partial fulfilment of the requirements

for the degree of Bachelor of Technology

by

Nidheesh Rajesh Panchal (2016UCP1008)

Siddhant Gupta (2016UCP1455)

Rahul Jangir (2016UCP1396)

1. Consistency of paragraph spacing is kept throughout. There paragraphs are pushed to another page as the new page had the starting line as some incomplete statement.

2. Heading styles and font styles for normal text is kept consistent. The captions for figures are checked and the font style is converted to normal text.

3. Figure and table numbering are checked and the numbers referred to in the text is corrected.

4. Captions are given same font style

5. Indentations are made uniform throughout the document.

6. Inferences are drawn from the results presented in the two methods discussed and checked for missing inferences in Experimental results too.

7. The small figures are made large so that the text is clearly visible.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

MALAVIYA NATIONAL INSTITUTE OF TECHNOLOGY

MAY 2020

Certificate

We,

NIDHEESH RAJESH PANCHAL (2016UCP1008)

SIDHHANT GUPTA (2016UCP1455)

RAHUL JANGIR (2016UCP1396)

Declare that this thesis titled, “Android Pin Cracking” and the work presented in it are our own. I confirm that:

- This project work was done wholly or mainly while in candidature for a B.Tech. degree in the department of computer science and engineering at Malaviya National Institute of Technology Jaipur (MNIT).

- Where any part of this thesis has previously been submitted for a degree or any other qualification at MNIT or any other institution, this has been clearly stated. Where we have consulted the published work of others, this is always clearly attributed. Where we have quoted from the work of others, the source is always given. With the exception of such quotations, this dissertation is entirely our own work.

- We have acknowledged all of the main sources of help.

Signed: _____

Date: _____

Dr Vijay Laxmi

Professor

May 2020 Department of Computer Science and Engineering

Malaviya National Institute of Technology Jaipur

Abstract

The project aims at predicting the PIN codes of an android phone that a user may unlock using a numeric PIN code. Sensors pre-installed on every smartphone work without any acknowledgement to the user and can fetch data continuously without raising suspicion. Using the orientation and movement data from such sensors, a person can detect any touch screen activity by observing a spike in the data from sensors that log the motion of the phone. Exploiting the fact that sensors like accelerometer, gyroscope, and gravity do not stop recording even when the phone is locked, we aim to crack the PIN of a mobile phone.

The data was collected using an app from multiple environments such as putting a phone on a table, operating the phone using one hand. Different sensors' data used and harvesting machine learning power, we aim to predict what part of the screen is being touched and, in turn, predict the PIN.

Acknowledgement

We would like to express our sincere gratitude to Dr Vijay Laxmi, our project supervisor, for her patient guidance, enthusiastic encouragement, and useful critiques of this work. A constant push and constructive feedbacks made this possible and helped us to have hands-on experience with the new technologies.

Nidheesh Rajesh Panchal (2016UCP1008)

Siddhant Gupta (2016UCP1455)

Rahul Jangir (2016UCP1396)

Contents

List of figures

List of tables

Chapter 1

1. Introduction

In this era of digitization, technology and the way of living is changing fast. Technology is now becoming a new usual way of living. About 5 years ago, in India, there was less awareness of the Internet and its uses, but today every other person has access to it. The problem is not about using this technology but not having enough awareness about the security loopholes exploited by malicious users.

Security and privacy are two major concerns in our day and age. As most of the things are moving online, the vulnerability increases with the ease of access and usage that it provides. We have online banking, study programs, online food ordering apps, government services, online payments. Moreover, the use of social media has increased with the usage of internet services. There is hardly any person who does not hold a social media account. After all these advances in technology, internet services, and applications, we still need more security and privacy. Even though security is also kept in mind while launching new services or products, the question still stands. Is that security sufficient?

1.1 Risks

We have all of our information, including our financial, personal photographs, personal text messages, location information, and more, stored on our phones. Whenever we connect to the internet, all such information is at stake. This information can be used for many malicious activities such as accessing our bank accounts, tracing our location, tracking our user history. They get to know what we have purchased and give personalized advertisements whenever you search for some product online or on any social media site. Hence, in this advancing age of smartphones, it is essential to know how safe your phone is along with your laptop.

1.2 Awareness

There are vulnerabilities to storing information on the internet. We all know that once something is up on the internet, it may always stay up there. There are very few people in the world who understand how the online systems work and how vulnerable it could be. What type of data can prove to be very harmful if put on the internet? There are people from this small percentage too who do not care about how other people use the data. This shows that even the aware users ignore the privacy terms and conditions and blindly click accept whenever they require to use a service. So, leave alone the unaware users who just follow these trends that are going on in the world.

1.3 Legalities of data access

There are legal ways your private information can be taken and maybe even sold to the data scientists who require such data or to the advertisements seller to get the target audience. There are many ways in which your privacy and security can be compromised. Accepting terms and conditions without reading is the first step towards vulnerability.

Hacking is one type of breach into your system from remote. It depends on how a loophole in the code of service can be used to the advantage and steal any information that the hacker seems fit to benefit from. This is an illegal manner of taking the information. This area itself has many types.

1.4 Strength of password

People use simple passwords or PINs so that they can remember them in the time of need. They may set the password or PIN as the birthdate or even a mobile number or license plate number of their vehicle. Thanks to all the social media sites, birthday information is readily available and can even be seen publicly. All these passwords can be easy to guess in a brute force method.

Nowadays, the systems have upgraded their password requirements, where they check the length of the password entered and the characters used in it. Longer the length of the password, the stronger it is. All secured accounts login asks the user to enter a password such that it contains at least one uppercase letter, one lowercase letter, a number, and a special character and keep the minimum length of the password required as 8. If anyone tries every combination possible to crack the password, he/she has to try over a billion passwords after knowing that the length of the password is 8.

1.5 Permissions

What if someone can take your password in a legal manner. Just like how any person simply clicks “OK” while installing an application on the phone and gives all the permission it asks for blindly. What if someone takes your media storage entirely online without you noticing. Ask yourself this question that how many times have you read the complete terms and conditions before clicking on “Accept” and give permissions to the camera, microphone, location, and storage to any app that you have installed.

Let’s say that a person has not given any permission to the application, but by coming into physical contact of the phone, he/she can unlock the phone to transfer any data he/she wants. This can be possible if the person knows the PIN code of the phone. Unlocking can be very tough if the person does not know the actual PIN as the PIN can be of different lengths. It may lock the phone for a specific amount of time until it allows the user to retry. But using all the possible PINs to unlock the phone is time-consuming, and the probability that you get the right PIN without permanently locking the phone is very low.

1.6 User experience

With the advancing in the virtual reality (VR) and the importance given to user experience with the phone, we usually see the functionalities like auto-rotate screen, pedometer being used by almost every other user, and if noticed, these functions are used without any permissions asked the user. When you enable auto-rotate screen, then it keeps track of the orientation of the phone in 3D space and rotates it in either landscape or portrait mode. Even while playing games like car racing, they, too, use these motion sensors, which requires no permissions to be asked and enhances the gaming experience. A smartphone that gives no permissions to any app can still be vulnerable to this type of attack that takes data from motion sensors of the phone.

1.7 Risks of motion sensors

Focusing specifically in the direction of exploiting the motion and position sensors of the android smartphones in the background, we aim to record the sensor data, using a gyroscope (angular velocity), accelerometer (linear acceleration) and gravity, from the phone and predict the numeric PIN used to lock the phone. These sensors require no permissions when accessed and can easily leak information to the background service. This background service can upload such sensitive data from the background itself to remote storage or can be taken in real-time database form. Internet connection to a mobile phone is a very common thing nowadays, and any of these remote accesses will not be a limitation to the attacker.

It is observed that whenever the screen is touched on specific portions, a spike is created in the motion and position sensors. Position and motion sensors tell us about the orientation and movement of the phone in 3D space. Through extracting such information of the motion of the phone when the screen is touched and released these patterns can be analyzed and used to know which part of the screen was touched, we can crack the PIN of a phone.

To do so, we log the data from sensors such as accelerometer, gyroscope, and gravity, to know the orientation and movement of the phone in 3D space. This raw data is first processed so that it is noise-free and contains all the required features to recognize the pattern. This processed file is then used as an input dataset to feed into a supervised multiclass machine learning (ML) model. As the numeric PINs contain numbers from (0-9), we will be using multiclass classification for 10 classes which represent the keys pressed, or in other words, tell which part of the screen was touched where that particular number key will be present when the user is entering the PIN.

Our method will not be limited to the length of the PIN and will be used to predict individual digit from the complete sensor data so that it is flexible to any PIN length. It can also be seen that the PIN is not only used for unlocking the phone but also for online transactions. There are PINs to be set for online payment apps and can be scaled up to be used for this kind of exploitation also.

We are going to introduce two methods in which we will be explaining how the data is collected and how the data is being processed before feeding it to the machine learning model as input. We have tried different configurations for each model used and have compared each processing method and sensor type data to eliminate the models that do not give accurate results or the sensor type, which will not be having the data which is required to get the correct predictions.

For the actual exploitation of the sensor data, we will be building a background service to allow remote access to all such data in such a way that the user is not aware of the background process that is running. The app will store the data locally and also upload it to cloud storage. These files are further processed in a similar manner in which the training data is processed. It is then given as input to the trained machine learning model and give out results.

We will be considering the results for 50 4-digit PINs and see by what probability we can predict individual digits of the PIN. We will also compare our results to the related work on this particular topic. 50 PINs contain 200 individual digits, so the probability of predicting on single digit can be calculated and compared with the brute force method of exploitation.

If it is possible to train machine learning models to predict which part of the screen is touched by the user to an extent, then it would mean that it could be extended further to predict the keyboard input also and, in turn, increase the vulnerability associated with it. This can increase the concerns related to security and can become a security risk level the same as the keylogger in a device.

With the increasing usage of smartphones for net banking and other secured services, this exploitation, if scaled to a level, can then become a high risk to all such applications as no password or OTP will then ever be hidden from the attacker. Even the two-step verification will stand as useless and ineffective against this type of attack.

One of the few possible ways to stop this vulnerability is by adding permissions to these motion sensors and prevent it from working in the background. This may affect the user experience on the phone as it will then start asking permission for almost every sensor on the phone and will display a notification of it being used all the time in the notification panel.

Chapter 2

2. Important Terms and Concepts

Added text

Before moving on to discuss our method, we would like to state some important terms and concepts. There are some traditional methods stated in this chapter and a few machine learning methods along with two related works.

2.1 Traditional methods for password cracking

There are many ways in which the password can be cracked. The passwords are always kept as an encrypted entity. There are a few techniques that can be used:

- Stealing: By looking at someone's computer while he/she is entering a password in the system. There may be a note written by a user to remember password which can be stolen. A different method can be to install a keylogger on a system and take that log to record the characters entered for the password.
- Phishing: When a user clicks on a fake website sent on a mail can lead them to a site which is not actually a secured one and simply takes the password in text form along with the username.
- User analysis: A user generally tends to set a password related to his birthday, license plate number of the vehicle he owns, name of pets, mobile number, spouse's name or even their child's name. All this information can be extracted from various social media platforms, and it is known as spidering.
- Brute-force attack: When an attacker tries each and every possible combination to crack the password. [20]

We can spot an example for Windows 7, where we could change the password for the admin using the loophole that was present in the startup repair. The non-admin user can run the startup repair and change the password by copying the "sethc" file. The unrestricted access to the file that is not supposed to be in the non-admin user partition is exploited to get admin access to the computer. This is just an example of the older windows version. [21]

2.2 ML to crack laptop passwords

In the early days, merely setting a strong password was enough to protect your system from a breach. Over time a stronger password has been asked to use, which includes the uppercase and the lowercase characters, special characters, and numbers. Due to this, many new techniques have been evolved to crack the passwords. Many researchers are working on finding new ways to crack the password of any system. Machine learning and deep learning has helped them to crack passwords like never before.

Stevens Institute of Technology, New Jersey and New York Institute of Technology have come together and built software based on machine learning techniques which cracks password. They managed to train their model with leaked data of passwords. And now, this model can crack the password without any prior knowledge of the user. This model is known as passGAN. Also, they have merged their model with HashCat, which increased the probability of guessing the correct password from 50% to 70%.

Many researchers have found that the noise of key stroking can be used to crack a password. For example, if someone is on a Skype call or any video or audio conferencing, the noise of a keystroke can be heard and used to train machine learning models to find how a particular key is pressed and what key is pressed. The noise of keystroke for a particular person is supposed to be always the same as he/she practices typing on the keyboard. And this practice sticks with a person. So while unlocking the system, the password will be typed in a similar manner as he is using the keyboard regularly and have entered passwords multiple times and have developed a habit of using it in a particular manner. These keystrokes will create noise, and finally, this noise can be used to find the password to the system. [19]

There is a similar concept that can be used to crack the password of a smartphone. Instead of using keystroke sounds, we will use how keys are pressed by the user and monitor the motion of the phone when the phone is held in hand. Motion sensors respond uniquely for every keypress, and this response can be used to train a machine learning model. This is the concept on which this project is based.

We will be using a supervised learning technique for machine learning where the input data contains the data (feature vectors) and the output corresponding to that data so that the model adjusts itself by looking at the output that it produces and comparing it with the output which is expected.

2.3 Importance of cracking passwords

Leaving aside the harms done by the malicious attacker, we look at the importance and need of such cracking techniques. If, for example, an extremely secured system is locked by a user who is not alive anymore and the password was only known to them. The data inside the system is too valuable and is not backed up anywhere; then, all these cracking methods are needed to retrieve the password to the system.

All this research work on the cracking of password helps us to develop maximum security for the data of our country's military, which may include very sensitive data such as missile launching codes. It may also help us to hack into an enemy's system and get an intelligence report to prevent harms done to the country.

2.4 Multiclass Classification Models

It is a classification model that labels the input and categorizes it under one class from the multiple classes that are defined. For example, there can be many weather conditions possible according to the data input, and each sample is labelled under one class.

We will be using the following classifiers in our project:

- Random Forest
- Logistic Regression
- Naïve Bayes Classifier
- KNN Classifier
- Decision Tree
- Support Vector Machines
- Multi-Layer Perceptron (MLP)

2.4.1 Multi-Layer Perceptron model (MLP)

The basic perceptron algorithm says that in the perceptron, we just multiply with weights and add bias and do this in one layer only. In MLP, there can be more than one linear layer (combinations of neurons), i.e., having hidden layers in the network of nodes. Here we will be using MLP from the sklearn library and Keras library naming the models as MLP and Deep Learning model, respectively. These two different libraries provide a wide range of activation functions, optimizers, and the different modifiable parameters, making them both important.

2.5 Multioutput Classification

As the name suggests, it used to get multiple outputs for a single input sample. Each output has its own set of properties, and each property may have multiple classes. It is like a combination of multilabel and multiclass classification. [13]

2.6 Related Work

Two of the well-known published papers on the topic of exploiting smartphone sensors for cracking a PIN are [15][17]. They have shown remarkable results in this research topic, and the methods used by both Berend et al. and Mehrnezhad et al. are different.

indentation corrected

2.6.1 Method used by Mehrnezhad et al.

The method used by them is that first, they recorded 50 fixed 4-digit PINs, which are very common in use. They used a browser (Chrome) on a Nexus 5 device to collect all such data. They allowed a single user to enter theses 50 PINs five times, and the experimental results are based on this data.

They did not restrict their users to hold the phone in any particular manner. They allowed them to use it in the most natural position that they like and feel comfortable with (using one or two hands for input).

They considered both time-domain and frequency-domain features to get the feature vectors for input. They considered four sensor data; accelerometer including gravity, accelerometer without gravity, orientation, and gyroscope. They removed the initial values before moving forward with feature extraction. They had 36 features in both frequency and time domain. They considered the energy of each sequence in both domains and added this to the 36 features to make it 60 features. They then combined each sensor type to every other sensor type, each with 3 elements. 18 new features were added to the previous 60 features. Their feature vector finally consisted of a total of 114 features. They used a supervised learning method using an artificial neural network (ANN). As input, ANN received a set of 114 features for each sample. They had 2488 input samples for ANN.

paragraph gaps made ocnsistent

They split the data into two, making the 70% data as the training data and rest 30% was further divided into two; testing and validating equally. They used a network of one hidden layer consisting of 1000 nodes using a scaled conjugate gradient (SCG) for updating weights.

They divided the data into two, namely single and multiple users. In the single-user mode, one participant had entered data through the browser, and in the first attempt, it could correctly predict the complete 4-digit PIN with 74.43% accuracy, and the accuracy gets better as the number of attempts increases. For the multiple-user, 10 participants add the data through the browser and fill the complete dataset by themselves. This achieved an accuracy of 70.75% in the first attempt and increasing with the number of attempts.

The classifier performed better when it was personalized. The attacker could guess the PIN from a set of 81 possible PINs with 71.82% probability of success. [15]

This result is from the method used by the published paper.

2.6.2 Method used by D. Berend et al.

In this method, instead of taking the complete 4-digit PIN sensor data as input to the ML model, they have decided to identify each digit of the sensor data individually. They take the data for the complete 4-digit PIN in one go and then separate out the relevant sensor data to classify each key press individually. This approach is very flexible as compared to the previous method, as the previous method has a limitation of the PIN length. If a model is trained with only 4-digit PIN, then it will be predicting only those PINs. Moreover, using 50 PINs out of the total possible combination of 10,000 PINs for training the model is not adequate.

Using an individual digit classifier makes it easier to get all the possible 10 classes of PIN digits. This can be used to predict PIN of any length. They, too, chose an LG Nexus 5 for data collection. They collected 70 4-digit PINs from users who were asked to enter these PINs 5 times. They had selected these PINs in such a way that every number (0-9) appears before and after each number. Other 50 random 4-digit PINs were entered for validating purpose.

They had restricted the user to use only the right-hand thumb to type the passwords at a consistent speed. Each sensor data had the time and the coordinate values corresponding to each timestamp (in milliseconds). They made the data uniform so that every millisecond has an entry in the data. They discarded the first four data values from the data so that the initial motion when the recoding starts is removed. Then, by taking an average of a few initial values and replaced all the values with the absolute difference.

They processed the file in such a way that the window of onKeyUp and onKeyDown is kept at the end and combined the accelerometer and gyroscope sensor type data.

They used Multi-Layer Perceptron (MLP) neural network with two hidden layers where the first layer has the same number nodes as the input vectors, and the second includes 10 nodes. They used the LBFGS optimizer. They used a total of 2500 files for training and tested with 116 test samples. [17]

paragraph spacing consistent

We have referenced these two works in our project and have also compared our results to them so that we know the position at which our work stands at every level. We have used some processing methods that are similar to [17] for the comparison and have also introduced new methods for the same. The actual exploitation of the PIN by extracting the data from a phone when it is locked is absent from these two papers. So, the results from that are not compared to any of these two papers.

When the key up and key down timestamps are not known to the attacker, and only has access to the raw data from the sensors, he/she needs to develop some other method to get a timestamp by merely looking at the graph. One such method is proposed ahead and concluded.

Chapter 3

3. Literature Survey

added text

In this chapter, we have discussed the password mechanism of Android and the type of sensors and that are present in a typical Android phone. The selection of correct sensor type is important, and thus it is necessary to know what type of motion is recorded by which sensor. It discusses two different methods of exploitation that are possible.

3.1 Android password mechanism

Password protection is very much necessary with the increasing amount of sensitive data on the phone. Every operating system has its own mechanism for it. 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 these android keys. As these keys are directly related to password and its security, it is difficult to crack an android lock screen password. Android not only provides hardware-backed storage of keys but also provides keymaster for security services. It provides an execution environment that is trusted and secure services (example strongbox).

3.1.2 User authentication

Android provides PIN, pattern, biometric as well as face unlock. Android keeps gatekeeper for PIN and pattern authentication and fingerprint for biometric authentication. In android devices that are using the Nougat Android version and above can use the biometric prompt for both fingerprint and biometric data. The authentication states of these components are communicated with the Keystore service.

3.1.3 Credentials enrolment

Whenever user boots device for the first time, i.e., after a factory reset, all authenticators like a key, biometric, or face are ready to receive credentials enrolment from a user. A user is required to provide these details with authenticators. The user has to provide at least 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 works as a token which binds user, and it's cryptography. SID binds to the user password cryptographically. Whenever a user authenticates successfully with Gatekeeper, Auth tokens are generated. These contain the SID of the user and help in a successful authentication. [18]

If a 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 that don't notify the user about their usage and can be accessed by any app or website (through a browser) silently.

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

3.2 Android Sensors

All Android phones, as a matter of fact, all smartphones have sensors that will help to measure some essential environmental conditions and motion of the phone to give enhanced user experience. They can monitor the 3D movement and position of the device.

The Android platform supports three broad categories of sensors:

- Motion
- Environment
- Position

3.2.1 Motion Sensor

It monitors movements such as tilt, shake, rotation, or swing. The movement can be caused by direct user input or by the physical environment in which the device is sitting. You can monitor motion relative to the device's frame of reference and movement relative to the world's frame of reference. [2]

3.2.2 Position Sensor

It lets you determine the position of the device in 3D space. It checks the orientation of the phone using the geomagnetic sensor and accelerometer sensor. The installation of an orientation sensor was stopped after Android 2.2 (API level 8), and the use of orientation sensor type to get sensor data stopped after Android 4.4W (API level 20). [3]

That is, instead of using raw data from a dedicated orientation sensor, its sensor type, we use a data combination from the accelerometer and geomagnetic sensor.

3.2.3 Accelerometer Sensor

An accelerometer is a sensor that measures the linear acceleration of a device in its instantaneous rest frame,[5] When the device is at rest, the accelerometer with gravity sensor type will measure the acceleration due to gravity along with the linear acceleration. Hence, it shows the Z value of the sensor as ($g \approx 9.81\ ms^{-1}$), whereas the accelerometer without gravity sensor type does not include this gravitational acceleration data and simply measures the linear acceleration in the world frame of reference.

3.2.4 Gyroscope Sensor

Gyroscope fetches the data that tells us about the angular velocity of the device on its axes, i.e., speed of rotation around its axes.

3.2.5 Gravity Sensor

The gravitational force has three vector components, in X, Y & Z directions. A gravity sensor senses these components of a smartphone and tells the alignment or orientation of the phone. It uses gravity acting on it. (Z=9.8 when at rest).

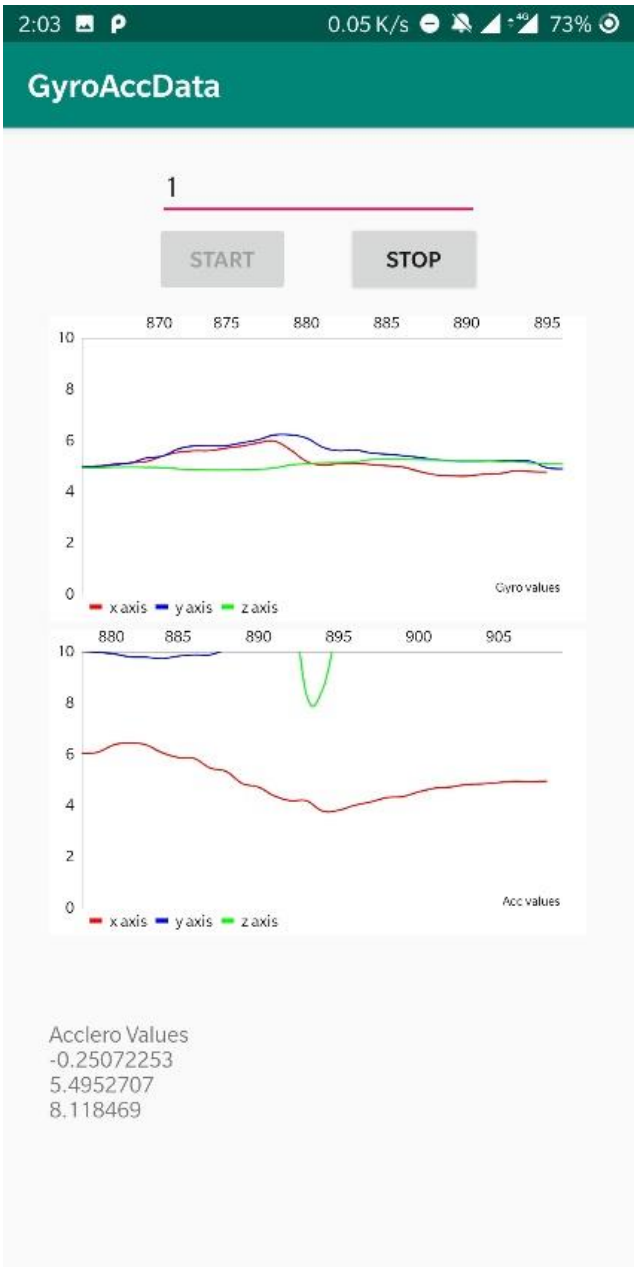
When a SensorEventListener is registered and used on an application, even after locking the phone while keeping the app open in the background, it could still log the sensor data without giving any warning or message to the user of the phone. But if the app is closed by the user, the services are killed to free the space. Therefore we will be using a background service to record the sensor data.

For receiving the data from these sensors, we can either use an application (background service) or use a Javascript API to get the sensor data to the server. For the android application, we can make use of the SensorEventListener by registering the listener to the sensors and saving them in a .csv file. The site implementation for getting sensor data on the Chrome browser is done by the “Generic Sensor API.” The data receiving rate (events rate) is low in the Generic Sensor API when compared to the application implementation for the same. The data is received at every 16ms for the site, and for the application, it varies according to the onSensorChangeEvent. This is when the frequency for the sensor is kept the highest; we can also change the frequency of data that is received but will use the highest frequency to get very detailed motion data of the phone.

3.3 0-permission sensors

Today we have various operating systems on our phones, which include 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.



scaled figure so that text is visible

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

Consistent figure and table captions.

added details of the frequency of sampling and delays

In Figure 1, the top graph plots Gyroscope values and the bottom graph plots Accelerometer values. All the sensors were kept at a maximum sampling frequency (with nearly 0s delay in the average number of samples obtained in one second) 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. The range of sensor delays between two SensorEvent for OnePlus 5T is as follows:

- Accelerometer with gravity: 2.5 – 1000 (ms)
- Accelerometer without gravity: 5 – 200 (ms)
- Gyroscope: 2.5 – 1000 (ms)
- Gravity: 5 – 200 (ms)

There can be different frequencies for sensors at which they record data depending on the specific hardware installed on a smartphone. The sensor types accelerometer with gravity and gyroscope are considered high-frequency sensors as compared to the other two.

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, and an app can use these sensors' data without any intimation to the user. This can potentially be exploited to leak sensitive information. In this report, we aim to demonstrate that the sensors' data can be used to steal the Personal Identification Number (PIN) of a user from his/her motion patterns.

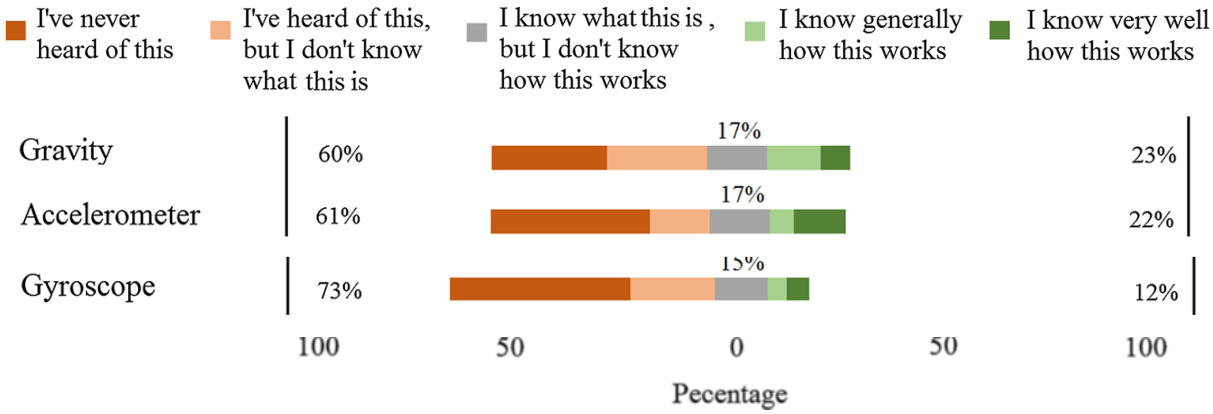


Figure 2: Sensors at its knowledge to user

Figure 2 is 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.

3.4 Using sensors for predicting touch

From Figure 1, we know that it is possible to see when the screen is being touched. Using the high-frequency sensors (accelerometer with gravity and gyroscope), 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 3).

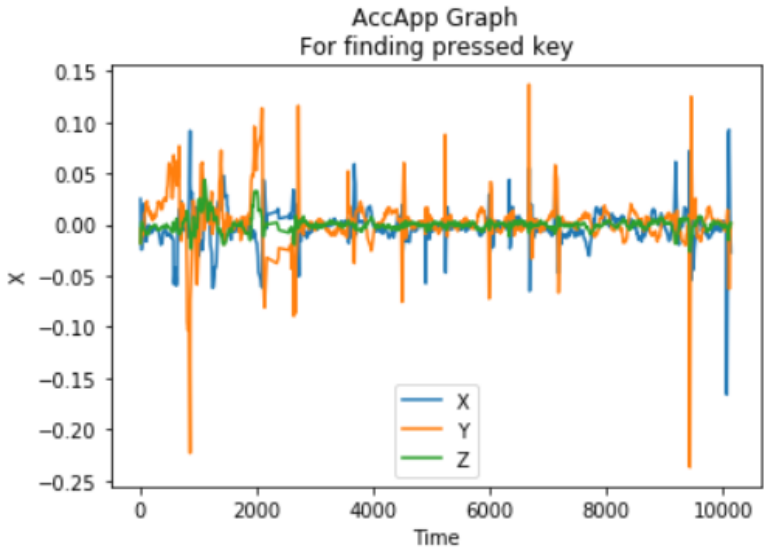


Figure 3: Unique spikes in the graph when different keys are pressed

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.

3.5 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.

3.5.1 Site

As seen in the implementation of the paper [15], they made 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]

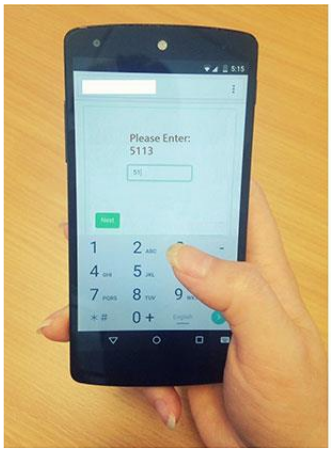


Figure 4: Site opened on a phone

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].

3.5.2 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 presses 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 5, the app also provides a numerical keypad for the user to enter PINs.

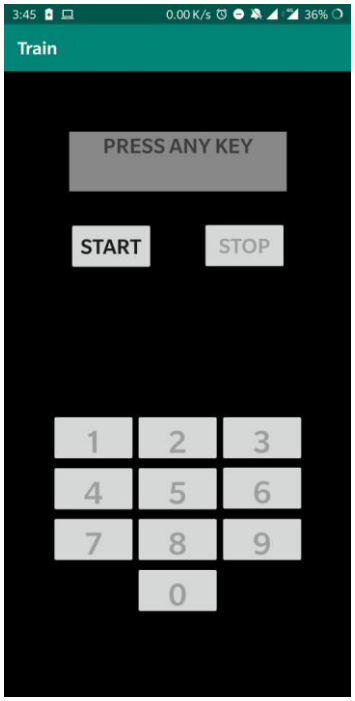


Figure 5: “Train” application that takes sensor data in the background

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 have four files, each which is stored on the external storage. A user presses the “stop” button, and all sensors are unregistered and stop data collection. After each key press, 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 collects data from the user using a similar approach from section 3.3.1. It is discussed in detail further in the report.

Chapter 4

4. Proposed Method

After observing the results from the experiments performed, we propose a method to identify individual digits of a PIN. We have decided to separate all the results in the order that it was evolved, namely Method-1 and Method-2.

4.1 Method-1

This method is the first experimenting step, where we tried a strictly random data collection method and used a few basic data processing methods to convert raw data to be input to the various machine learning models.

4.1.1 Data collection

To start with training a Machine Learning model, first, we begin with a fundamental step of data collection from a single user. Using the application "Train," as mentioned in section 3.3.2, a user is made to either sit or stand and hold the phone in right hand in one identical position every time. The app collects four types of sensor data

- Accelerometer with gravity (Acc_w): It records the device motion in 3D space and shows the rate of change in each of the three axes merged with gravity. (TYPE_ACCELEROMETER). When the phone is stationary (z=9.8). (z value represents the change in z-axis along with gravity acting on that axis)
- Accelerometer without gravity (Acc_wo): It records the device motion in 3D space and shows the rate of change in each of the three axes without merging gravity. (TYPE_LINEAR_ACCELERATION). When the phone is stationary (z=0). (z value represents the change in the z-axis)
- Gyroscope (Gyro): Rate of rotation in all three axes.
- Gravity (Grav): Force of activity in all three axes.

When the "Start" button is pressed in the app, the sensors are registered, and they start collecting data until a number key is pressed by the user, as shown on the screen. As soon as the key is pressed, a gap of 1 second is kept to allow the keypress data to be well distributed. After the gap, a file is saved in (.csv) file format from all four sensor types to their corresponding folders. In this method, the app shows random numbers on the screen. There are 40 such numbers displayed in a single session. It is made sure that all numbers (0-9) are pressed four times each. After completion, the user is asked to press the "stop" button.

Multiple sessions were taken on different days or a different time of day to allow a slight variation in the initial orientation of the phone. The gravity sensors and accelerometer with gravity sensor type keep such a record of the orientation. 560 files were collected by the "in-hand" method.

To know if there is a possibility that the digits can be identified if there was minimal phone movement, we collected data by keeping the phone on the table (with protective case), and another 400 files were collected.

4.1.2 Pre-processing

The raw data collected from the application contains Time and X, Y, Z coordinate values. The file name is coded in such a way that it includes the timestamp of onKeyDown, onKeyUp and the number key that was pressed. We have used different methods of pre-processing the data and compared the same with different models of classification. The pre-processing methods used are as follows.

- Windowing
- Three values
- Three values from uniform data

The complete training dataset of processed 560 files from in-hand data and 400 files from on-table data was saved in a single .csv file for input to the classification models.

4.1.2.1 Windowing

Taking a window of 25 values so that the initial and final values that are noisy and not helpful for classification and will simply confuse a model, are removed. So now the window of 25 values contains some initial values, the middle index contains the window of (onKeyDown to onKeyUp) data, and the rest of the data is appended according to the space left in this window of 25 values. After stripping the “Time” column from the data, these values are converted to a single row as feature vectors in a dataset, so that there are (25*3) feature vectors.

4.1.2.2 Three values

After taking the same window frame of 25 values as windowing, now we consider only three values, i.e., first value, mid-value of the key down and key up timestamp window, and the final value of the total window. It is then converted to a single row and put in the dataset as feature vectors after stripping the “Time” column. In the end, this pre-processing method gives (3*3) feature vectors.

4.1.2.3 Three values from uniform data

The data collected by the sensors do not have an entry for every millisecond. They log data only when an onSensorEvent is generated so that only a change in values is recorded. The sensor frequency is not that fast to get an event and do the job in the required listeners.

So first, we make the data uniform in the intervals of 5 milliseconds where we fill the gap by taking the nearest timestamp to the interval and copying the corresponding data to this particular entry. After making the data uniform, the same processing is done as in the previous section.

4.1.3 Classification models

The choosing of the classification models initially was based on the general usage of these models. Following are the models that we have trained:

- Decision Tree
- Random Forest
- Logistic Regression
- Naïve Bayes
- KNN
- SVM
- Multioutput

All the above-mentioned models were used from the sklearn library for python. The complete dataset was split into two, making the 20% data as the testing data for each model.

The parameters of the classification models were changed in such a way that it gives the best maximum accuracy from all the processed files.

4.1.3.1 Decision Tree

Thee different processed files were added as input, and the results from each were compared. The top 3 results can be seen in Table 1.

Table 1: Decision Tree results

Processed type	Sensor type	Accuracy (%)
4.1.2.2	Gyro	31.2
4.1.2.1	Acc_w	29.78
4.1.2.1	Gyro	27.74

We can see that the primary windowing method is able to give good accuracy for a greater number of times than the one with three values only. The gyroscope sensor type shows better accuracy than an accelerometer. After changing the parameters multiple times to get the best results, as shown in Table 1, the model definition is shown in Figure 6.

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=15, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

Figure 6: Decision Tree parameters

This result did not allow us to make a clear decision on which processing method is best to use as it changes after the top result. Same is the case with the sensor type.

4.1.3.2 Random Forest

The top 3 results can be seen in the table below.

Table 2: Random Forest results

Processed type	Sensor type	Accuracy (%)
4.1.2.1	Gyro	40.7
4.1.2.2	Gyro	33.33
4.1.2.3	Gyro	32.62

In the random forest, we can see that only the gyroscope sensor type gives top accuracies. The maximum is given by the window processed file. Random forest model shows clear domination of Gyroscope sensor over other sensor types, but the processing methods that appear on the top 3 list makes the selection for processing method difficult. The final parameters set for this output is as shown below.

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=10, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=10,
                       n_jobs=None, oob_score=False, random_state=0, verbose=0,
                       warm_start=False)
```

Figure 7: Random Forest parameters

4.1.3.3 Logistic Regression

The top 3 results can be seen in the table below.

Table 3: Logistic Regression results

Processed type	Sensor type	Accuracy (%)
4.1.2.3	Gyro	42.47
4.1.2.1	Gyro	41.13
4.1.2.2	Acc_w	39.71

The logistic regression model has shown a variety in the processed files that gives the best accuracies. The accelerometer appears at the bottom with 39.71% with the windowing processing type, and the top place is still taken by the gyroscope. The gyroscope has started to appear like a good choice of sensor type, but the variance in the processing type is not yet resolved. The final parameters set for this output is as shown below.

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=0, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

Figure 8: Logistic Regression parameters

4.1.3.4 Naïve Bayes

The top 3 results can be seen in the table below.

Table 4: Naive Bayes results

Processed type	Sensor type	Accuracy (%)
4.1.2.3	Gyro	41.13
4.1.2.1	Gyro	36.28
4.1.2.2	Gyro	33.62

Here the gyroscope sensor type dominates other sensor types, and the top result is from the three values uniform time processed file. The final parameters set for this output is as shown below.

```
GaussianNB(priors=None, var_smoothing=1e-09)
```

Figure 9: Naive Bayes parameters

4.1.3.5 KNN (K-Nearest Neighbours)

The top 3 results can be seen in the table below.

Table 5: KNN results

Processed type	Sensor type	Accuracy (%)
4.1.2.1	Gyro	33.62
4.1.2.2	Acc_w	33.33
4.1.2.3	Gyro	31.85

The windowing processed file comes up on top of the table with 33.62% accuracy. The gyroscope again appears in the top 3 more than once. KNN clustering model is not able to give accuracies as high as given by Logistic Regression. The model is not able to cluster the data points properly and fails. The final parameters set for this output is as shown below.

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=7, p=2,
                     weights='uniform')
```

Figure 10: KNN parameters

4.1.3.6 SVM (Support Vector Machine)

The top 3 results can be seen in the table below.

Table 6: SVM results

Processed type	Sensor type	Accuracy (%)
4.1.2.1	Gyro	42.47
4.1.2.2	Gyro	38.5
4.1.2.1	Acc_w	37.6

The windowing processed files from both accelerometer with gravity and gyroscope appear in the top 3, having a difference of about 5%. Data plotted on higher dimensions from the gyroscope sensor type is separable with higher accuracy than on lower dimensions (3). The final parameters set for this output is as shown below.

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
    max_iter=-1, probability=True, random_state=None, shrinking=True, tol=0.001,
    verbose=False)
```

Figure 11: SVM parameters

4.1.3.7 Multioutput

It uses Random Forest in the multi-output classification. The random forest parameters are kept the same as used earlier, which are set to give the best results. In the multi-output classification, there will be more than one input label and output labels. So, we arranged the input labels such that the first index of the input of 4 contains the actual key that is pressed, and in the next three, we put in the keys that are near this key and have chances to be predicted as the output. For example, if the actual key label is 1, we add 2, 5, and 4 in the list of 4 input labels. The output is shown in the experimental results section.

We can see that none of the above tables shows the gravity sensor type and was always appearing at the bottom of the comparison table. The gravity sensor frequency was not high enough to detect minimal movement. We stopped using the gravity sensor for comparison.

4.2 Method-2

After observing results from the various processing methods and classification models used in Method-1 and taking inspiration from the related works, we developed the Method-2 for data collection and changed the processing methods to convert raw data to input. The classification models used in this method include two complex neural networks.

4.2.1 Data collection

In this method, we have approached the data collection technique in a different manner. Previously we were taking random numbers and asking the user to hold the phone in an identical position. This way of holding the phone limits the machine learning model from learning a particular holding technique, so we asked the user to hold the phone as they usually do but still use the right-hand thumb for input.
Instead of displaying random numbers on the screen, we were now displaying the numbers in such a way that every number appears once, before and after every number. For example, if 2 is pressed after 1, then 1 should be displayed after 2 also. 90 such keys were displayed in a single session, and a total of 2700 files were stored. There is an equal key label distribution in these files.

It uses the same “Train” app for data collection that we modified for implementing this method. The user is asked to keep hovering his/her thumb over the key that is pressed last until a new number is displayed. This helps to mimic the motion of the phone when the actual PIN is entered.
This time we did not collect gravity sensor data and used only three sensor type values:

- Accelerometer without gravity (Acc_wo)
- Accelerometer with gravity (Acc_w)
- Gyroscope (Gyro)

4.2.2 Pre-processing

After observing the raw data values closely, it was seen that the onKeyUp and onKeyDown window could be as large as 100ms. So, if we take a total window size of 25, then it will be too short of having all the relevant data required. We increased the total window size that will be taken from the raw data to 200 and also compared the results for window size 400.

The pre-processing methods used are as follows:

- Windowing
- Window with uniform time
- Window with uniform time and timestamp (TS) at the end
- Window with uniform time and removing initial values
- Window with uniform time, removing initial values, timestamp (TS) at the end

We know that accelerometer measures linear acceleration and gyroscope is the rate of rotation, using any one of the sensor types may not give the best results. So we combine two sensor type data, i.e., the two accelerometer variants are combined with the gyroscope. We have also kept the individual sensor type processed file for comparison.

When the two sensor type data is combined, different combinations of window size is used (200 or 400) for both accelerometer and gyroscope. The combination is done in such a way that the accelerometer variant data is first converted to a single row in the same way we did before, and then the gyroscope type data is appended to the same row adding to the feature vectors. For example, the total feature vectors for 200 window size of the accelerometer variant and 200 window size of the gyroscope will be (200*3) + (200*3) = 1200.

4.2.2.1 Windowing

Taking a window of 200 and 400 values. The total window contains some initial values. The middle index contains the window of (onKeyDown to onKeyUp) data, and the rest of the data is appended according to the space left in the total window. After stripping the “Time” column from the data, these values are converted to a single row as feature vectors in a dataset.

4.2.2.2 Window with uniform time

As we know that the sensor does not log for each, and every millisecond, the data is not uniform. To make it uniform, this time, we keep an entry for every millisecond, and the missing entry is copied from the previous entry that is present. When making the raw data uniform, the same windowing is done, as explained in 4.2.2.1.

4.2.2.3 Window with uniform time and TS at the end

In this pr-processing method, the same steps are followed as from the 4.2.2.2. The only difference is that instead of putting the onKeyDown timestamp in the middle index of the total window we will keep the onKeyUp index at the last index so that the total window contains the complete path of a finger reaching the key but not of leaving the key which might be irrelevant

4.2.2.4 Window with uniform time and removing initial values

This pre-processing is similar to 4.2.2.2. But before taking the total window out, we first take the starting 200 values and discard it from the raw data file and then with first average the first 300 values so as to get the initial position. This initial position is then subtracted from the remaining data. It is used with a logic that the initial orientation of the phone should not affect the data arrangement of the same digit file if the phone is held differently. First, we used the method of merely removing the initial first value from the complete file without using absolute, then referring to [17] and looking at their success with such processing; we continued with this method of taking the average instead.

4.2.2.5 Window with uniform time, removing initial values, TS at the end

Taking the intermediate processed file from 4.2.2.4, and keeping the window of onKeyUp and onKeyDown at the end of the total window. It is similar to [17] (differs in window size) used for comparison of our results.

4.2.3 Classification model

We have eliminated some models from the previous methods as they were not showing any potential for improvement and give better results than the overall test results. We added 2 neural network models, MLP, and Deep Learning model. We are using different names for them as MLP is used from a sklearn library, and the Deep Learning model is used from Keras library.

Following are the models that we have used:

- Logistic Regression
- SVM
- MLP
- Deep learning

The complete dataset was split into two, making the 25% data as the testing data for each model. The parameters of the classification models were changed in such a way that it gives the best maximum accuracy from all the processed files.

4.2.3.1 Logistic Regression

Logistic regression model basically allows us to classify binary results i.e., true/ false. The logistic regression can be extended to give multiclass output by assigning individual probabilities to each class and then output the class that gets the highest accuracy.

The different processed files were added as input, and the results from each were compared. The top 5 results can be seen in the table below.

Table 7: Method-2 Logistic Regression results

Processed type	Sensor type (size)			Accuracy (%)
	Acc_wo	Acc_w	Gyro	
4.2.2.1	200	-	200	64.29
4.2.2.2	200	-	400	62.97
4.2.2.1	200	-	400	62.13
4.2.2.2	400	-	400	62.09
4.2.2.2	200	-	200	60

The table shows that the accelerometer with gravity has no contribution to give the top 5 accuracies and the increased window size and combination of two sensor type has increased the accuracy of the model. The final parameters set for this output is as shown below.

```
LogisticRegressionCV(Cs=10, class_weight=None, cv=3, dual=False,
                     fit_intercept=True, intercept_scaling=1.0, l1_ratios=None,
                     max_iter=1000, multi_class='auto', n_jobs=None,
                     penalty='l2', random_state=0, refit=True, scoring=None,
                     solver='lbfgs', tol=0.01, verbose=0)
```

Figure 12: Method-2 Logistic Regression parameters

4.2.3.2 SVM (Support Vector Machine)

Support vector machines plot the data points in high dimension (depends on the number of feature vectors) and then try to find a hyperplane that separates those points such that the few selected vectors belonging to either class fall on one side of the hyperplane and are equidistant to the hyperplane.

We observed results by keeping the same configuration for the model as used before, and it was giving the best accuracies as seen below.

Table 8: Method-2 SVM results

Processed type	Sensor type (size)			Accuracy (%)
	Acc_wo	Acc_w	Gyro	
4.2.2.2	400	-	400	58.55
4.2.2.1	-	-	400	57.45
4.2.2.2	-	400	400	57.14
4.2.2.2	200	-	400	56.93
4.2.2.1	-	200	200	56.45

From the table, we can see that the accuracies have undoubtedly increased for SVM, but when compared to the logistic model, it still has a difference of 6-7%. SVM shows poor performance as compared to Logistic Regression model for test data.

4.2.3.3 MLP (Multi-Layer Perceptron)

The MLP model is a neural network model with one input layer, one output layer and can consist of one or more hidden layers. The number of nodes in the input layer is same as the number of feature vectors in the training dataset and the number of nodes in the hidden layers can be set by passing a tuple parameter to the MLP function developed by sklearn library. All the nodes in one layer are connected to all the nodes in the next layer in the sequence and the weights are adjusted for each connection made to the node according to the error that is backpropagated. The learning rate decides how fast a model can learn and modify the weights. For our model, the hidden layer sizes were kept as the total window size of the input combining two sensor types. The top 5 results can be seen in Table 9.

Table 9: MLP results

Processed type	Sensor type (size)			Accuracy (%)
	Acc_wo	Acc_w	Gyro	
4.2.2.1	200	-	400	66.38
4.2.2.1	200	-	200	65.98
4.2.2.1	-	200	200	63.92
4.2.2.1	-	200	400	63.22
4.2.2.2	-	200	400	62.68

We see that the window processing method has dominated the table with the accelerometer without the gravity sensor type combined with the gyroscope. It gives an outstanding accuracy of 66.38% and shows a high potential for better training for a larger dataset. The final parameters set for this output is as shown below.

```
MLPClassifier(activation='identity', alpha=0.0001, batch_size='auto',
              beta_1=0.9, beta_2=0.999, early_stopping=True, epsilon=1e-08,
              hidden_layer_sizes=(400, 10), learning_rate='constant',
              learning_rate_init=0.001, max_fun=15000, max_iter=1000,
              momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True,
              power_t=0.5, random_state=None, shuffle=True, solver='lbfgs',
              tol=0.01, validation_fraction=0.1, verbose=False,
              warm_start=True)
```

Figure 13: MLP parameters

4.2.3.4 Deep learning

The deep learning model is just another name used for the neural network implementation in the Keras library. There are wide range of activation functions available for each hidden layer and output layer. After testing a number of possible combinations of different hidden layer sizes, activation functions, and optimizers, we also added regularisation to avoid overfitting and give good results. The final parameters set for this model are as follows.

```
{"class_name": "Sequential", "config": {"name": "sequential_1", "layers": [{"class_name": "Dense",
    "config": {"name": "dense_1", "trainable": true, "batch_input_shape": [null, 1200], "dtype": "
    float32", "units": 4, "activation": "relu", "use_bias": true, "kernel_initializer": {"
    class_name": "VarianceScaling", "config": {"scale": 1.0, "mode": "fan_avg", "distribution": "
    uniform", "seed": null}}, "bias_initializer": {"class_name": "Zeros", "config": {}}, "
    kernel_regularizer": null, "bias_regularizer": null, "activity_regularizer": null, "
    kernel_constraint": null, "bias_constraint": null}}, {"class_name": "Dense", "config": {"name":
     "dense_2", "trainable": true, "dtype": "float32", "units": 10, "activation": "softmax", "
    use_bias": true, "kernel_initializer": {"class_name": "VarianceScaling", "config": {"scale":
    1.0, "mode": "fan_avg", "distribution": "uniform", "seed": null}}, "bias_initializer": {"
    class_name": "Zeros", "config": {}}, "kernel_regularizer": null, "bias_regularizer": null, "
    activity_regularizer": null, "kernel_constraint": null, "bias_constraint": null}}]}, "
    keras_version": "2.3.1", "backend": "tensorflow"}
```


Figure 14: Deep learning layers and parameters

4.3 Exploitation App

After observing the results from all the above experiments to predict the individual PIN digit, it was time to test it on the field. There are two significant ways by which this can be done, either by using a site or an app. It was observed that after losing focus from the tab which runs the exploitation service, it stops recording the sensor data, so it is not usable, and another reason to not use the site is that it does not have the same frequency of getting sensor data as that of background service. For the app, too, there is this limitation that the user should install the app in the first place.

The functionality of the app must include that it should run a service in the background such that it will not alert the user of its existence. The app “Pin the PIN” was developed, keeping this in mind. Due to the updates after the Nougat Android version, it is not possible to keep the service running in the background without keeping a notification in the notification panel informing the user about its existence. So, all the phones having the Nougat Android version or below are vulnerable to this exploit once the app is installed on the phone.

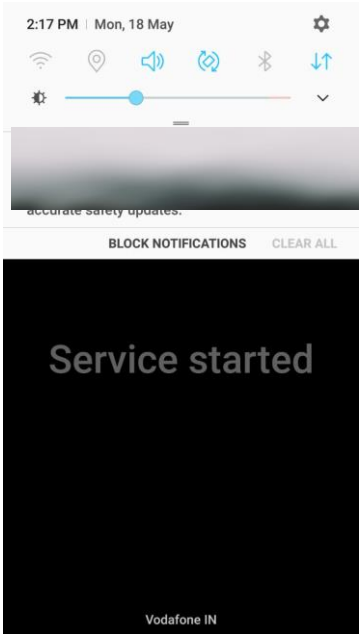


Figure 15: No notification for Android Nougat

The above screenshot is of a Samsung Galaxy A7 running Android version Nougat, and thus the service running does not get displayed in the notification panel and will not be killed.

With new Android 10 in the market, there are 39.2% of Android phones that use Nougat or below (till April 2020). To overcome this low percentage of target phones, what can be done is, for the Android versions higher than Nougat, the app service will simply sit there in the notification panel saying “Service running in the background,” and this notification is given a top priority. If the notification is not displayed, then the service is killed after some time to save power. Considering that the service must run all the time, even if the app is opened once, the notification is to be created by the service. The app can be modified a bit to resemble any app that runs in the background and shows different functionalities from what it announces to be doing. Taking an example of the Notification logger app, which has to work 24/7 on the phone to log the user notifications, the user is okay with it running in the background as he/she trusts the app, which can be one possible modification of the app.

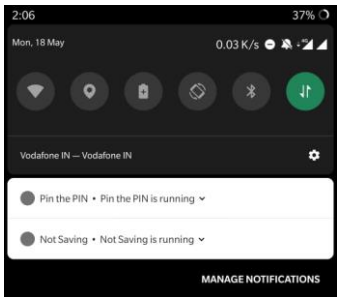


Figure 16: "Service running in background" notification on Android Pie

The above screenshot is from OnePlus 5T Using Android version Pie (9.0). As it can be seen that the notification is the same as the notification logger app, “Not saving.”

What the app does is, when the phone is locked and the screen is off, the data of the sensor is not recorded. As soon as the phone screen is turned on by the user, it starts logging the sensor values. After the phone is unlocked, the recording stops and saves the file in the internal app storage so that it is not available as a media file on the storage, and along with that, the files are uploaded to the Google Firebase storage so that it is accessible from anywhere. It requires an internet connection to upload the data files, but in today’s world of increased usage of the internet, there are hardly any phones that do not use the internet regularly. The files which are uploaded to the Google Firebase storage are then downloaded for further processing.

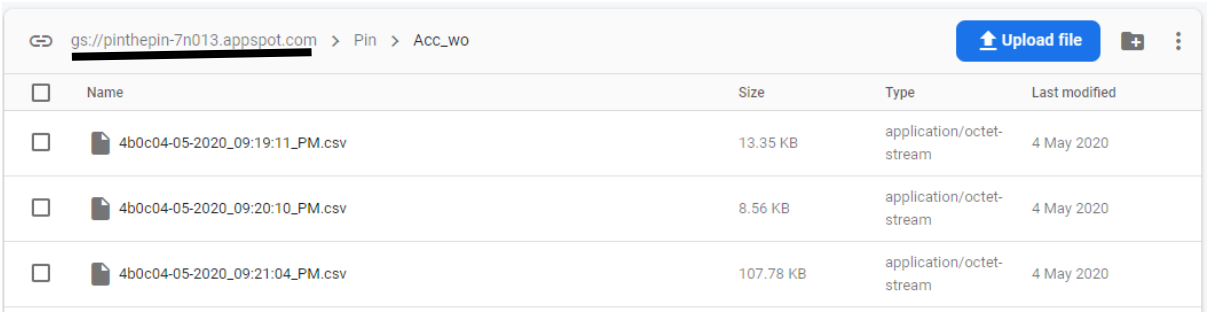


Figure 17: Google Firebase storage where data is uploaded

First, a graph is plotted for the file, which contains the accelerometer without gravity sensor data as it shows a proper graph of the linear motion and hence interpretable. It could be seen that the PIN of length four was entered by observing the five spikes in the graph. The first spike is the one when the user picks up a phone and swipes up on screen to get the PIN keypad and the other four spikes being the corresponding keys that were pressed. There is a little gap between them all, and are split in that way. Now we need to get the timestamp of the keypress by using only the graph of the file. All the four split data files having an accelerometer without gravity sensor data are subjected to local minima search, and the lowest value of the Z coordinate is chosen as the timestamp of the key press and is appended in the file name.

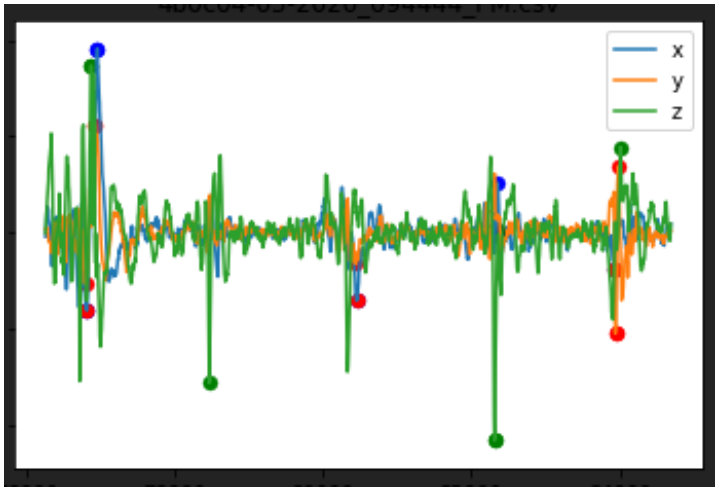


Figure 18: 5 spikes seen in the data for one unlock

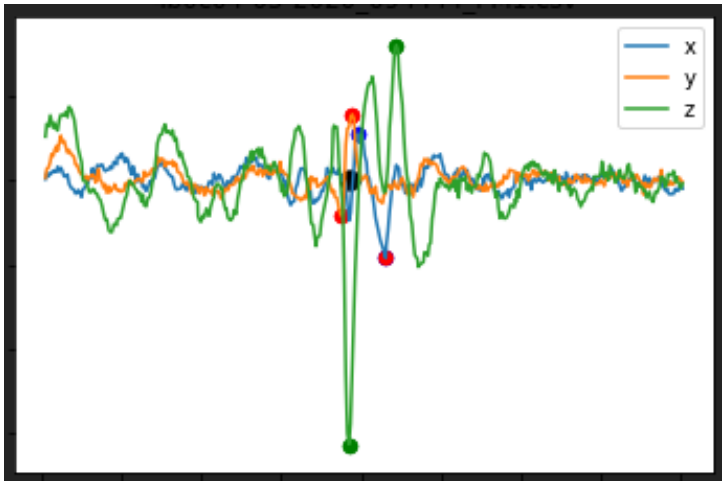


Figure 19: Split and get the timestamp for processing (black dot shows timestamp)

The file is then further processed according to the model which we are using to predict the keys, using the same processing method, we get a file with four rows (4-digit PIN). Each key press gives 2-guess data for the key that might have been pressed. All the results are described in Experimental Results.

Chapter 5

5. Experimental Results

The final results from the two methods are presented. This allows a comparison of the different classification models in each method. The tabular data is formed by picking up the best values for the corresponding classification model.

5.1 Method 1

It was observed that motion detected by the gravity sensor did not show remarkable changes in sensor data values. Hence it did not contribute to give accurate prediction (16%). The sensor data from the sensor type accelerometer without gravity also did not provide accurate results (12.3%). We can observe that gyroscope and accelerometer with gravity are the winners among all the other types of sensors. It can be seen in Table 3. It shows maximum accuracy among all kinds of processing methods.

Table 10: Comparison of classification models in method 1

Type of data	Random Forest (%)		Decision Tree (%)		Logistic (%)		Naïve Bayes (%)		KNN (%)		SVM (%)	
	Acc	Gyro	Acc	Gyro	Acc	Gyro	Acc	Gyro	Acc	Gyro	Acc	Gyro
On-table	37.5	42.5	20	34	19	19	22.5	23.75	35	28.75	17.5	35
In-hand	38.93	40.7	27.65	31.2	36.87	41.13	26.54	33.62	32.74	42.47	33.62	42.47
Hybrid	34.71	31.6	29.87	29.46	19.91	26.14	12.43	22.27	36.78	39.89	26.94	27.46

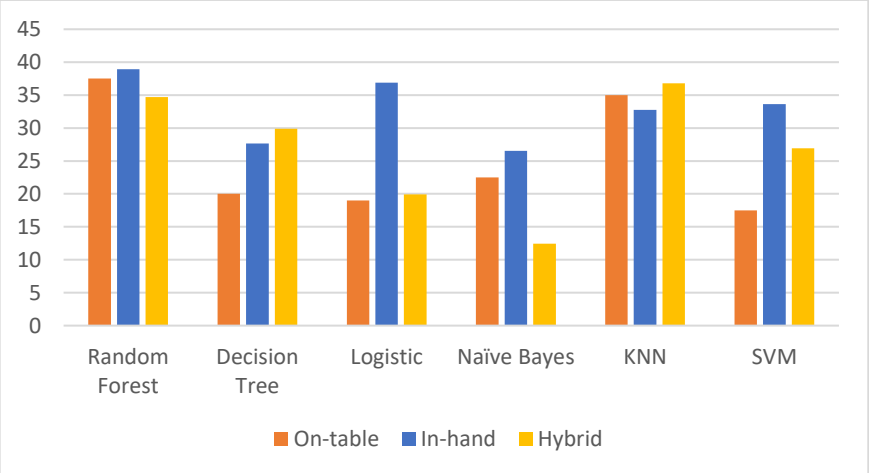


Figure 20: Method-1 accelerometer results

Figure 20 shows the comparison on a bar chart for the accelerometer sensor type, and we can see that the KNN and Random forest has performed very well in individual on-table and in-hand data, and the same goes for the hybrid data. Logistic performs well for in-hand data as compared to other types of data. Naïve Bayes and decision tree comparatively show lower potential than other models. SVM shows good results for the in-hand type of data.

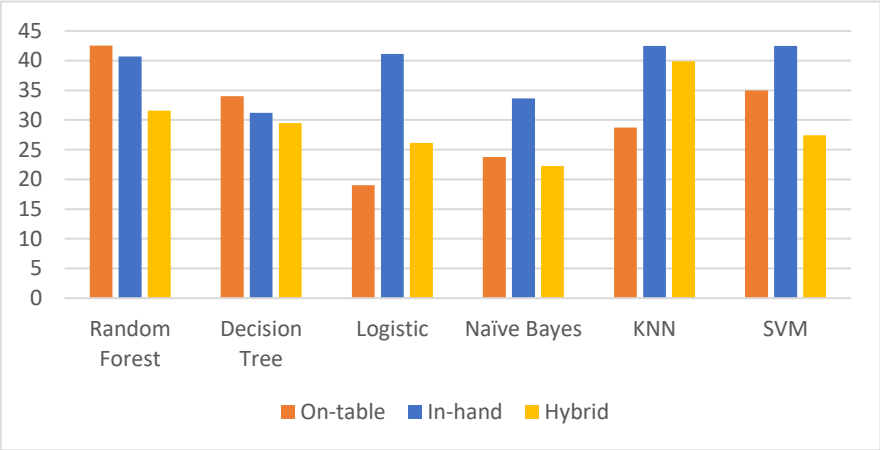


Figure 21: Method-1 gyroscope results

In Figure 21, we have compared all the accuracies from the gyroscope sensor type. The highest we can see is 42.5% from the random forest, and random forest shows a similar result for in-hand data type too. The decision tree maintains the consistency among the three data types, but they give lower accuracies. Logistic rose to 41.13% for in-hand data, and the other data types are still not giving good accuracies. Naïve Bayes' performance improved by almost 7%, but still, it is not up to the mark that it can be used further. KNN and SVM showed good results for in-hand data, but KNN has the highest accuracy for the hybrid type of data. Both gyroscope and accelerometer data appear to be important for the model to be trained best.

Table 11: Multioutput classification results

	Multioutput			
Type of data	Acc (%)		Gyro (%)	
	First 2	List of 4	First 2	List of 4
On-table	41	62	44	61
In-hand	49.64	72.34	59.57	73.75
Hybrid	43.56	61.82	51.45	67.63

In Table 11, the “first 2” column is the one that shows the output from index 1 and 2 from the output list of 4. If any of the two numbers match the first 2, it will be added to the accuracy. “List 4” is the complete list that of 4 outputs, and if any number matches the 4 outputs, then it is added to contribute to the accuracy. Even after comparing 2 possible outputs, the result did not rise as expected, that is why this model was eliminated.

In Table 3, it is seen that gyroscope is giving the best results up to 42.47% accuracy of a single-digit prediction when the phone is in hand showing that gyroscope is a sensor that should be given more weight if it is to be merged with another sensor type.

In Table 4, we can see that from the list of 4 output, the actual key matching either first or the second output can be maximum of 59.57% accurate by keeping the phone in hand and using only a gyroscope sensor. From the list of 4 output, the actual key present in that list, the maximum accuracy achieved is 73.75 using phone in-hand, and gyroscope is the sensor being used.

5.2 Method 2

In the second method, we have used different Classifiers and have discarded the previously used models that were not giving good results. The following table shows the maximum accuracies after comparison (the complete data is too big to be displayed). Using 2700 files in total and splitting it to get 25% of test data for validation.

Table 12: Method 2 test accuracies

Model	Type of processing			Accuracy (%)
	Acc w/o grav (size)	Gyro (size)	Type	
Logistic Regression	200	200	4.2.2.1	64.29
SVM	400	400	4.2.2.2	58.55
MLP	200	400	4.2.2.1	66.38
Deep learning model	200	200	4.2.2.1	64.99

Here, table 12 shows the window sizes used and the type of processing that shows the best accuracies from the complete comparison. All the other processing file types were eliminated. Then the single processing method that was giving good accuracies for 2 of the models (Logistic and MLP) was taken up for exploitation, i.e., accelerometer without gravity and Gyroscope sensor data with 200 window size of each and combined in a row to form feature vectors.

5.3 Pin the PIN

Taking 50 random 4-digit PINs to see how well the prediction works for the actual exploitation. The models are trained with a complete dataset of 2700 files and used for predictions. The following table shows the accuracy of predicting a single digit from all the keys pressed for those 50 PINs (50*4=200 digits) and the count of the correct digits predicted from a 4-digit PIN. The second guess of the digits surely boosts up the accuracy by at least 10% for each model.

(G1 → Guess 1) (G2 → Guess 2).

Table 13: Count of correct digits and accuracy

Model	1 digit		2 digits		3 digits		4 digits		Accuracies	
	G1	G2	G1	G2	G1	G2	G1	G2	G1	G2
Logistic	20	18	15	17	3	8	0	2	29.5	42
SVM	21	19	11	16	1	9	0	0	22	39
MLP	24	18	11	21	1	6	0	1	24.5	41
Deep Learning	11	23	2	4	0	3	0	0	7.5	20

The 1 digit, 2 digits, and so on represents the number of digits that are predicted correctly from the 4-digit PIN. The accuracies calculated are for each individual digit that is predicted correctly from 200. Logistic regression seems to come up as the best model out of all the models used. The next in line is MLP, with SVM right behind it. The deep learning model failed to perform as well as it did in the train test split because of the timestamp difference, which affects this model. The completely new data, when tested against this deep learning model, drops the accuracy drastically. Lack of training data can also be one of the reasons.

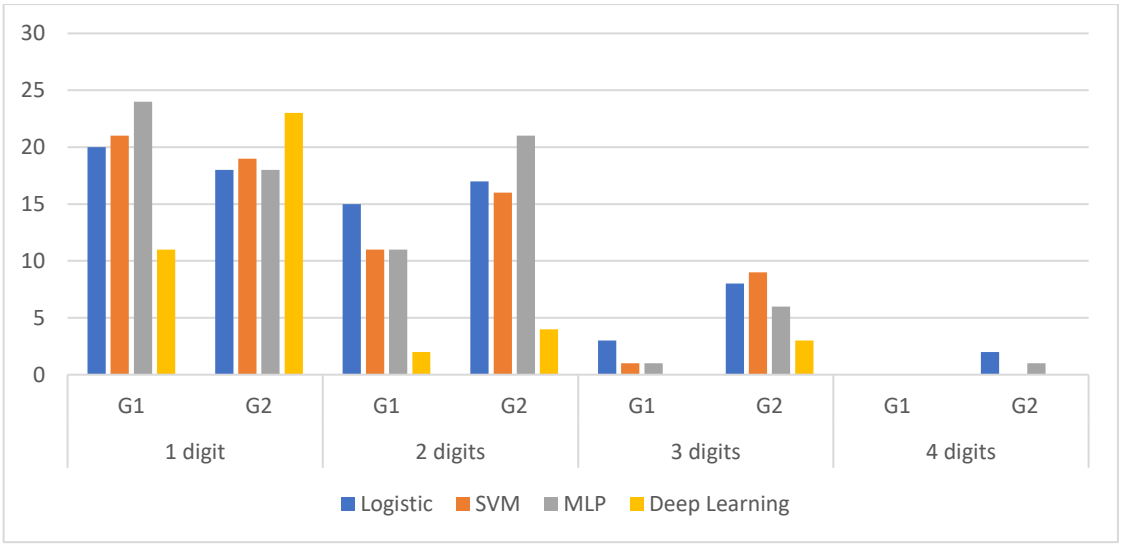


Figure 22: PIN prediction results (2500 files)

From Figure 22, we can see that the best performance for 1-digit prediction is by MLP, it drops for 2-digits prediction but still shows higher accuracy for the second guess. The logistic model performs well consistently. No model was able to predict all the 4 digits correct in the first guess, but MLP and the logistic model were able to do it in the second guess.

After extending the dataset to 6450 files, the count of the correct digits predicted is displayed in Table 14. We can see that the performance has improved by adding more amount of data to the training dataset. However, logistic performed the same with an increased amount of training data. SVM and deep learning models showed a significant rise in the accuracies. MLP showed no more than a 2% rise.

Table 14: Effect of extension of the dataset

Model	1 digit		2 digits		3 digits		4 digits		Accuracies	
	G1	G2	G1	G2	G1	G2	G1	G2	G1	G2
Logistic	26	12	13	29	1	6	1	1	29.5	46
SVM	19	17	9	16	7	7	0	4	29	43
MLP	19	19	13	20	2	8	0	0	25.5	41.5
Deep Learning	20	21	3	8	0	4	0	0	13	24.5

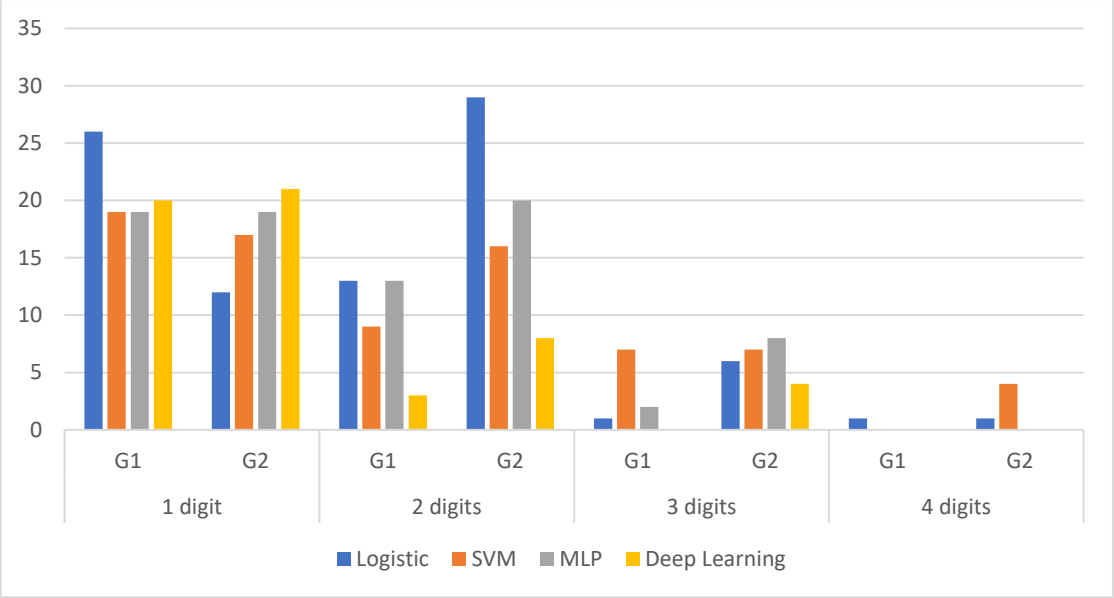


Figure 23: PIN prediction results (6450 files)

The main attraction of Figure 23 is that the logistic model was able to predict all the 4 digits correct for 1 PIN after the addition of new data. In the second guess, SVM surpasses logistic to predict 4 out of 50 PINs correct. We can say that the addition of new data can improve the models.

Chapter 6

6. Conclusions

Security and privacy are major concerns for the users when using any electronic device. We have explored how the sensors are used in the smartphones and the permissions and security related to these sensors. There were no permissions asked when using any of the motion sensors on the phone. This vulnerability was exploited to know about the user input patterns to get hold of his/her PIN for the phone. The targeted smartphone operating system is Android, as a majority of the people use Android phones. This does not mean that iOS provides any special permission requests before using these sensors. The only difference is that of how the background processes are handled by them.

By exploiting the sensors used on smartphones, it is possible to develop a single-digit classification machine learning model to predict the PIN entered by a user to unlock the phone. Different types of pre-processing methods and different classification models were used in method 1, where maximum accuracy achieved for on-hand data collection method is 42.5% using a random forest classification model, in-hand data collection method is 42.47% using a support vector machine model, and for merged data, it is 39.89% using K-Nearest Neighbours.

In method-2, we eliminated all the models that were not showing good accuracies from method-1. We increased the window size for the input file so that it contains all the required data of the path followed by the phone when it is touched. We can observe that the combination of both sensor type, accelerometer without gravity, and gyroscope with a window of 200 each give a good accuracy for Logistic Regression (64.29%) and Deep learning model (64.99%). Since getting a window of 400ms for a digit in the real-world scenario is not possible every time as the speed of typing may be high for a PIN as it is entered very frequently, the processing of file with larger window size is eliminated.

We can conclude that the PIN lock of a phone is crackable using these motion sensors to an extent. Out of 50 4-digit PINs, we were able to predict at least one digit correct for 38 4-digit PINs. The prediction accuracy of 28.36% for 200 digits from the Logistic Regression model is significantly higher than the primary cracking method of trying out every combination.

On comparing our results with [15] and [17], it can be observed that the model accuracies are not coming up to the same mark due to the different methods of data collection (71.82% and 83.7%). However, the exploitation of sensor data and their results are absent in [15] and [17]. Therefore, the comparison cannot be made on that account. The further exploitation of PIN from the phone and processing the file using only the graph with no proper time stamp reduces the accuracy of the already trained model of ours. The deep learning model did not perform well with the fresh new data because of the small size of the dataset.

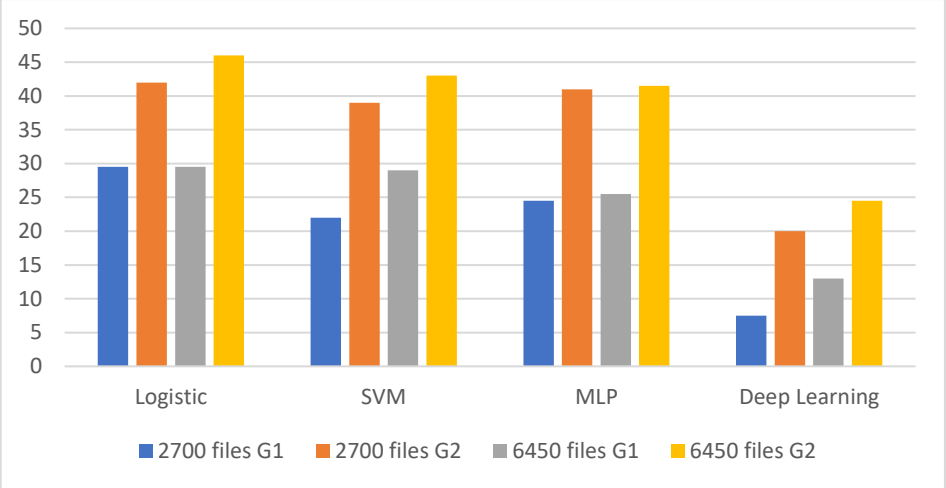


Figure 24: Comparison after adding files to the dataset

Observing the graph in the figure above, we can say that, by increasing the size of the dataset for training, we can improve predicting the power of a model. We can see a significant rise in accuracy in the Deep learning model and support vector machines.

As the accelerometer without gravity and Gyroscope sensors are not affected by the initial position of the phone in 3D space, the training dataset can be extended by profiling different ways of holding the phone alone. This method is flexible to accommodate longer PIN length (greater than 4) and can also be scaled up to be used not only to extract PIN used for locking the phone but to any internet banking apps and websites.

There are certain limitations attached to this, such as the person needs to install this app and open it once in order to start the background service. The app should provide functionalities that user trusts, and allows it to be run in the background. The internet connection to the phone is a must to upload the raw sensor data files to be accessed remotely.

References

[1] Sensors Overview | Android Developers. https://stuff.mit.edu/afs/sipb/project/android/docs/guide/topics/sensors/sensors_overview.html

[2] Motion sensors | Android Developers. https://developer.android.com/guide/topics/sensors/sensors_motion

[3] Position sensors | Android Developers. https://developer.android.com/guide/topics/sensors/sensors_position

[4] Tinder, Richard F. (2007). Relativistic Flight Mechanics and Space Travel: A Primer for Students, Engineers and Scientists. Morgan & Claypool Publishers. p. 33. ISBN 978-1-59829-130-8. Extract of page 33

[5] Rindler, W. (2013). Essential Relativity: Special, General, and Cosmological (illustrated ed.). Springer. p. 61. ISBN 978-1-4757-1135-6. Extract of page 61

[6] Corke, Peter (2017). Robotics, Vision and Control: Fundamental Algorithms In MATLAB (second, completely revised, extended and updated ed.). Springer. p. 83. ISBN 978-3-319-54413-7. Extract of page 83

[7] 3.2.4.3.2. sklearn.ensemble.RandomForestRegressor — scikit .... https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

[8] https://en.wikipedia.org/wiki/Logistic_regression

[9] Biopython - Machine Learning - Tutorialspoint. https://www.tutorialspoint.com/biopython/biopython_machine_learning.htm

[10] https://towardsdatascience.com/comparative-study-on-classic-machine-learning-algorithms-24f9ff6ab222

[11] Decision Tree - GeeksforGeeks. https://www.geeksforgeeks.org/decision-tree/

[12] Our objective is to find a plane that has the maximum https://www.coursehero.com/file/p27f3qtq/Our-objective-is-to-find-a-plane-that-has-the-maximum-margin-ie-the-maximum/

[13] 1.12. Multiclass and multilabel algorithms — scikit-learn .... https://scikit-learn.org/stable/modules/multiclass.html

[14] https://developer.android.com/docs

[15] Mehrnezhad, M., Toreini, E., Shahandashti, S.F. et al. Stealing PINs via mobile sensors: actual risk versus user perception. Int. J. Inf. Secur. 17, 291–313 (2018). https://doi.org/10.1007/s10207-017-0369-x

[16] Narain, S., Sanatinia, A., Noubir, G.: Single-stroke language-agnostic keylogging using stereo-microphones and domain-specific machine learning. In: Proceedings of the 2014 ACM Conference on Security and Privacy in Wireless and Mobile Networks, WiSec'14, pp. 201–212. ACM, New York (2014)

[17] Berend, David & Bhasin, Shivam & Jungk, Bernhard. (2018). There Goes Your PIN: Exploiting Smartphone Sensor Fusion Under Single and Cross User Setting. ARES 2018: Proceedings of the 13th International Conference on Availability, Reliability and Security. 1-10. 10.1145/3230833.3232826.

[18] Authentication - https://source.android.com/security/authentication

[19] New Password Guessing Attacks Use Machine Learning: How to Protect Against Them- https://www.apriorit.com/dev-blog/528-password-attacks-use-machine-learning

[20] Password Cracking and Countermeasures in Computer Security: A Survey - https://www.researchgate.net/publication/268978314_Password_Cracking_and_Countermeasures_in_Computer_Security_A_Survey

[21] Become Admin - https://null-byte.wonderhowto.com/how-to/hack-windows-7-become-admin-0160151/