An Approach to Software Development for Continuous Authentication of Smart Wearable Device Users

by

Tamalika Mukherjee

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved May 2017 by the
Graduate Supervisory Committee:

Sik-Sang Yau, Chair
Gail-Joon Ahn
Hasan Davulcu

ARIZONA STATE UNIVERSITY

August 2017

ABSTRACT

With the recent expansion in the use of wearable technology, a large number of users access personal data with these smart devices. The consumer market of wearables includes smartwatches, health and fitness bands, and gesture control armbands. These smart devices enable users to communicate with each other, control other devices, relax and work out more effectively. As part of their functionality, these devices store, transmit, and/or process sensitive user personal data, perhaps biological and location data, making them an abundant source of confidential user information. Thus, prevention of unauthorized access to wearables is necessary. In fact, it is important to effectively authenticate users to prevent intentional misuse or alteration of individual data. Current authentication methods for the legitimate users of smart wearable devices utilize passcodes, and graphical pattern based locks. These methods have the following problems: (1) passcodes can be stolen or copied, (2) they depend on conscious user inputs, which can be undesirable to a user, (3) they authenticate the user only at the beginning of the usage session, and (4) they do not consider user behavior or they do not adapt to evolving user behavior.

In this thesis, an approach is presented for developing software for continuous authentication of the legitimate user of a smart wearable device. With this approach, the legitimate user of a smart wearable device can be authenticated based on the user's behavioral biometrics in the form of motion gestures extracted from the embedded sensors of the smart wearable device. The continuous authentication of this approach is accomplished by adapting the authentication to user's gesture pattern changes. This approach is demonstrated by using two comprehensive datasets generated by two research groups, and it is shown that this approach achieves better performance than existing methods.

Index terms: continuous user authentication; smart wearable devices; behavioral biometrics; adaptive authentication.

DEDICATION

This thesis is dedicated to my parents.

It would not have been possible without you two!

ACKNOWLEDGMENTS

First, I would like to express my sincere gratitude to my adviser Professor Stephen S. Yau for letting me be a part of his research laboratory. I am thankful for his support, patience, motivation and for the immense knowledge that he shared with me. His guidance helped me a lot and I learned a lot from him.

Besides my adviser, I would like to thank Professors Gail-Joon Ahn and Hasan Davulcu for their insightful comments, and suggestions on my thesis and for taking the time to be members of my Supervisory Committee.

I also want to thank my laboratory mates Dr. Arun Balaji Buduru, Mr. Yaozhong Song, and Mr. Vineet Mishra for their valuable inputs.

Last but not the least; I would like to thank my dear friends Himanshu Waidya, Aditya Bivalkar, Aditya Dabak and Kedar Pitke for their constant encouragement, and support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Overview

Smart wearable devices (also referred to as wearable technology or wearables) have been touted as one of the fastest growing applications of the Internet of Things (IoT) environment [1]. As per a recent report [2], the annual worldwide shipment of wearables is projected to reach 214 million units in 2019 – a 250% growth from 2015. Wearables are sensor-equipped smart devices designed to be worn external to the body [3] or be embedded in clothing [4]. The consumer market of wearables includes products like the Apple Smart Watch [3], Fitbit [5], Myo gesture control armbands [6]. Figure 1 shows some of the popular smart wearable devices (or device types) available in the market today.

In the recent years, smart wearable devices have experienced rapid growth and wide scale adoption largely through affordable smart watches and fitness bands. Such is their popularity among users that an estimated nine out of ten smart phone sellers have come up with their own wearable device product or are about to launch one [36]. These smart devices provide novel ways to interact with users in the Internet of Things (IoT) environment. For instance, ubiquitous computing allows smart watch based wearables to enable users to avail limited features of a smart phone without the need to take out their phones. Users can communicate with each other, access emails, text messages, set reminders etc., just as they would with a smart phone. Besides, these smart devices very conveniently help measure user activities without requiring active user intervention. In the process, they motivate users to work out more

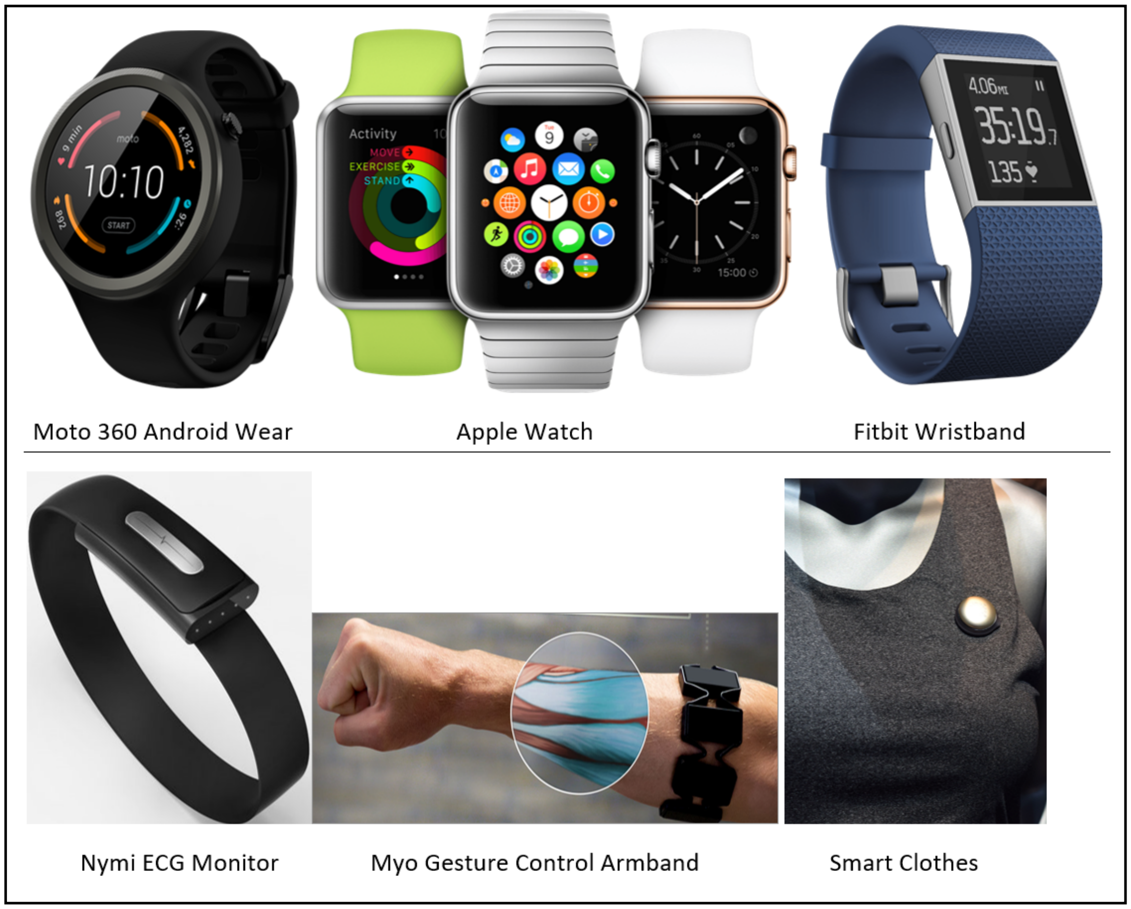


Fig 1: Popular Smart Wearable Devices

effectively, sleep well, essentially helping them adopt a healthier lifestyle. Along the same lines, smart clothes [4] plan to help users measure intensity of work out, check the number of calories lost during activities like running, exercising etc. Gesture control armbands like Myo [6] on the other hand help control other electronic devices like a TV or a laptop computer with motion gestures, from a distance. The sensor readings from Myo determine action performed based on which another device is controlled from a distance. In a nutshell, wearables offer fast, convenient and affordable technology that you wear thus enabling users to utilize its features hands-free.

Wearables are slowly foot stepping in the world of healthcare. Nymi wristband [55] measures heartbeat of the wearer with the help of embedded ECG sensors. Such devices have a huge potential of turning the healthcare industry by making it possible to provide patients with facilities like personalized medicines and better disease diagnosis [7, 37], by monitoring patient activities and vital statistics.

Although wearables offer quality services and convenience, they introduce new security, privacy and confidentiality preservation challenges. Wearables are generally used as personal electronic devices, because of which users store or access personal data with these devices. Consequently, as part of their functionality, wearables store, process and/or transmit confidential information such as personal health data, perhaps biometric, daily activities, location and/or communication data of a user. The nature and convenience of these devices make it easy to collect sensitive personal data unobtrusively and continuously. The need to protect such devices from unauthorized access is thus critical to their existence. Furthermore, wearables often operate by connecting to a secondary device (e.g. smart phone) via WiFi or Bluetooth, essentially connecting to other smart devices in the Internet of Things (IoT) environment, thus putting at risk other connected devices as well at the time of an attack. For example, research show that future wearables could be used as password managers that mediate access to other smart devices and online user accounts [52]. Wearables can also be used to access information such as personal bank account information [8], unlock cars, hotel or house doors, make payments etc. The association of wearables with such critical applications demand better security, protection and privacy preservation mechanisms. With a large majority of wearables entering the field of healthcare and personalized medicines, beyond privacy and confidentiality, information assurance and security must also encompass the critical aspects of safety and well-being of the individual. Malicious modification of user data may directly affect the operation and functionality of the device. An effective user authentication scheme is thus necessary to prevent unauthorized access, restrict information leak and to uplift consumer trust which would in turn help wearables achieve their true potential.

Current authentication techniques rely on the use of passwords, PINs or graphical pattern based locks. Adapting traditional authentication methods on wearables is especially challenging because most wearables lack an appropriate input interface to bolster reliable and secure entry of passwords. In fact, many wearables lack a UI or are too small to display a complex keypad (e.g. Myo armbands [6]). Figure 2 shows the Myo Gesture Control armband. Furthermore, an attacker can easily forge passwords or PINs. Another disadvantage of using such authentication methods is that they do not continuously monitor the user and would only authenticate the user at the beginning of the usage session [9]. In addition, users often dislike such explicit means of authentication as they disrupt their daily activities. Reportedly, 34% of US users do not employ even basic lock screen patterns to secure their smart device [10].

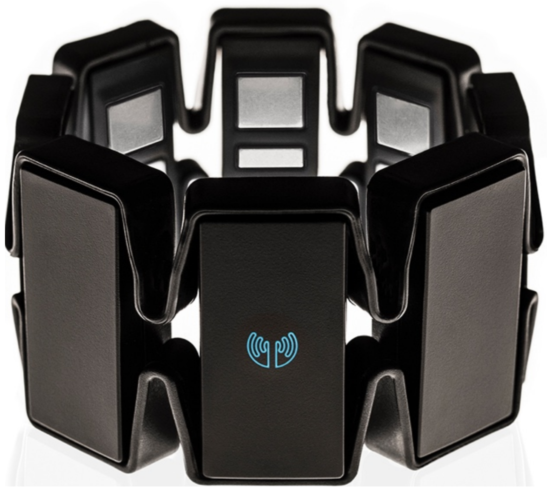


Fig 2: Myo Gesture Control Armband

Hence, an authentication mechanism that continuously and unobtrusively monitors and verifies the legitimate user is required. A continuous authentication scheme would verify the legitimate user throughout the usage session. On one hand, it is challenging to authenticate a wearable device user; on the other hand, these devices carry a multitude of cheap built-in sensors, which are capable of measuring user movements easily. In this thesis, we present a novel behavioral biometric based authentication approach utilizing user motion gesture data. Behavioral biometrics measure the consistency and uniqueness of behavioral characteristics of a user. Wearables are designed to capture biometric data continuously and uninterruptedly with the help of embedded sensors - we utilize this natural feature of wearables in our approach. Motion gesture data is collected through a wearable device via the built-in accelerometer and gyroscope sensors, and a user profile of the legitimate user is built. We use the accelerometer and the gyroscope sensors because complete motion information can be captured with a combination of these two sensors, which helps better model the user motion gesture patterns. In addition, these sensors do not require user permission to work. Based on the user model that is generated during the training phase, the legitimate user is validated during the verification phase. This approach is especially useful since the user wears a wearable device throughout the usage session facilitating continuous authentication. We implement scalable gradient boosting with decision trees to learn the user features. The effectiveness of our approach is demonstrated by using two comprehensive datasets. It is observed that our approach attains better performance than existing methods. An average accuracy of 98% and an average EER of less than 1% has been achieved. Up to perfect accuracy with no false positives has also been achieved by implementing our approach.

We also present an adaptive approach to update the generated model to capture user gesture pattern evolution. With the presented method, the original model is updated to incorporate information from the newly made available data. With ‘new data’ we refer to the data that is freshly generated and has not been previously used for training. This process of continuous learning takes up as many resources and as much time as it is required to just train on the ‘new’ data. It does not retrain again on all the available data. With ‘retraining’, we refer to training on the data on which the machine learning model has been trained in the past. Since, our approach trains on new data and does not exhibit retraining on previously trained data, it saves additional cost of retraining, since training models is essentially a costly event in terms of computational time and resources.

Our approach thus has the following important features:

(1) Utilizing behavioral features makes our approach more user-centric.

(2) The user of a smart wearable is authenticated continuously throughout the usage session.

(3) Conscious user inputs are not needed to carry out the authentication process.

(4) The adaptive scheme in our approach improves the performance of our approach when the user behavior evolves, and hence makes our approach more user-centric.

1.2 Organization of Thesis

The organization of the thesis is as follows: After discussing the current state of the art in Chapter 2, our overall approach is presented in Chapter 3. This chapter also layouts the threat model and our assumptions. Chapters 4 – 8 discuss the steps of our approach in details. Our experimental results and the evaluation of our approach is presented in Chapter 9. We discuss the conclusion and a probable future work for our current approach in Chapter 10. A list of references is provided next following Chapter 10.

CHAPTER 2

CURRENT STATE OF THE ART

Smart devices currently utilize authentication methods that are based on *knowledge-factors,* such as personal identification numbers (PINs), alphanumeric passwords, and graphical pattern based locks [9]. Figure 3 displays the passcode lock screen as it appears on an Apple Watch and the graphical pattern based lock screen as it appears on an Android Wear. In this scheme of authentication, the user generally unlocks the smart device by entering a known passcode, which is based on *what you know* or are *knowledge-based* [9]. However, this method can be broken if passcodes are lost, stolen or forged [11]. Furthermore, it has been demonstrated that only half as many smart device users use PINs [11] and 34% of US based users do not use even basic lock patterns on their smart devices [10]. This is since password-based identification methods are distracting to a common user. In addition, users tend to choose simpler password combinations mainly because of the undesirable load of recollecting complex mix of characters. A study [22] shows that out of more than 200,000 four-digit numeric passwords, 10 password designs made up 15% of the total, while two main password designs included designs as simple as `1234' and `0000'. In addition, 4-digit and 8-digit PINs have been shown to be broken within 40 minutes and 4 months respectively using existing techniques [38]. These methods are also vulnerable to shoulder surfing and smudge attacks [12, 13]. A smudge attack is a method by which oily residues or *smudges* from a touch pad can be used to detect passcodes or graphical pattern based locks. The simplicity and easy guess ability of passcodes or PINs, lack of wide-scale use [10], coupled with authentication only at the beginning of the usage session may cause bigger security breaches with such methods.

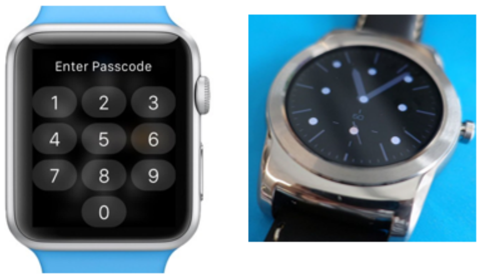


Fig 3: Passcode Lock (Apple) and Graphical Pattern Based Lock (Android)

Recently, two-factor authentication frameworks have been adopted. These require the user to input a one-time sign-in code besides entering the original password in general. This is considered a more reliable method of authentication since it combines the two verification components: *what you know* (secret key) and *what you have* (an alternative smart device) [9]. Figure 4 provides a pictorial representation

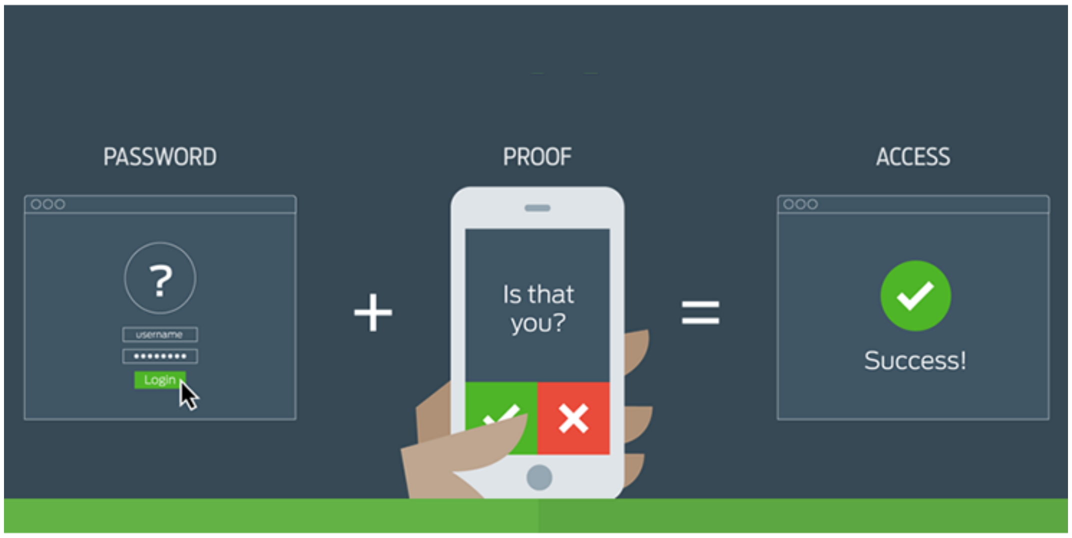


Fig 4: Two-Factor Authentication

of the method of two-factor authentication. In addition, a one-time password is considered a more secure alternative since it is valid for a short duration of time and thus has a lesser chance of being stolen or copied. However, the two-factor authentication scheme would fail in the event of theft attack, if the adversary assumes possession of the alternative smart device.

Traditional authentication methods are difficult to implement in wearables because of the small size of the input interface. Many wearables (e.g., Myo armbands [6]) do not even have an input interface to implement password based authentication methods. Majority of portable personal wearable devices like the popular smart watches have a small input interface and it is inconvenient to type in complex PINs. In fact, such devices may not have as much surface area necessary for the user interface to incorporate a complex keypad. A way out is to employ biometric based authentication methods, which authenticate the user, based on *who she is*. Biometrics can be broadly categorized into two types: Physiological and behavioral based biometrics, which are discussed next. Figure 5 shows the classification of the different authentication strategies observed in the current state of the art.

Physiological biometric verification utilizes the uniqueness of physiological characteristics, for example, fingerprint, iris patterns [29], and face [30]. Recently, physiological biometric based access control has been incorporated via fingerprint and face recognition technologies respectively by Apple and Google on their smart devices [14]. However, fingerprint recognition technology relies heavily on state of the art hardware and specialized sensors [15]. Besides, attackers can capture fingerprints from public events [16]. Whereas facial recognition technology depends on using a camera to authenticate a legitimate user. The face recognition technology measures facial features of a legitimate user during the enrollment phase. During the verification phase, the face to be identified is compared with the enrolled facial image using an

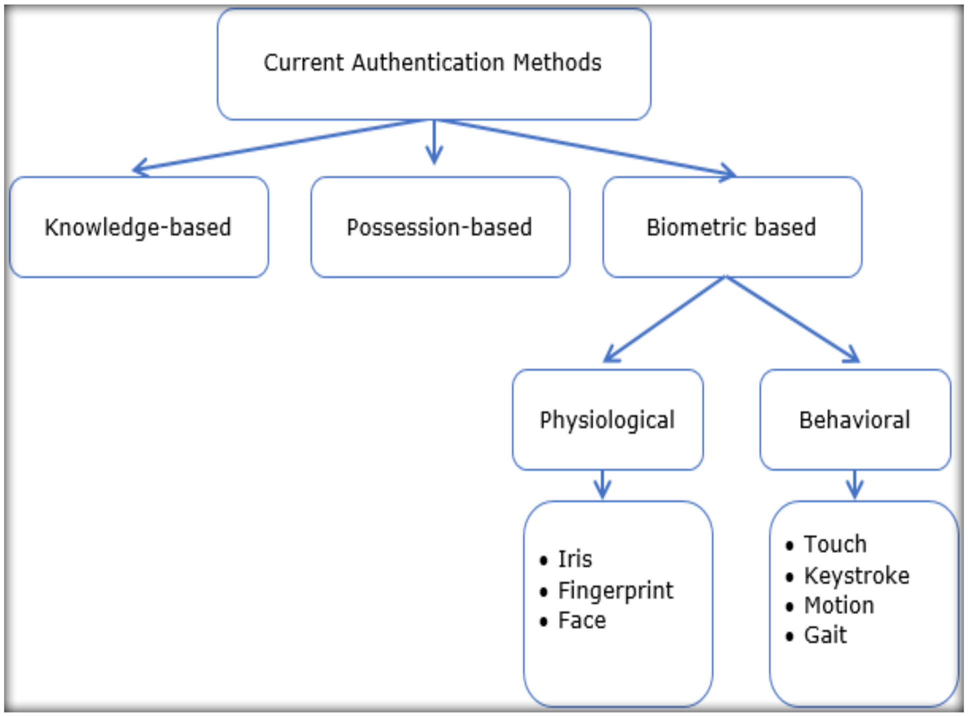


Fig 5: A Summary of Current User Authentication Methods for Smart Devices.

algorithm. Facial recognition has false rejection issues when individuals change hairstyle, shave facial hair or wear glasses or sunglasses [56]. In addition, face recognition fails to differentiate between identical twins [56]. A 2010 study shows that only 27% of the users would like to use facial recognition as a means of authentication owing to privacy concerns [17]. Google glass was disliked because of its facial recognition features. Consequently, Google dropped facial recognition from Glass [53]. In fact, the installation of additional hardware for physiological biometric based recognition systems increases cost and may not be feasible for lightweight, portable, low-cost wearable devices.

A disadvantage of the aforementioned modes of authentication is that they depend on conscious user inputs, which are likely to cause an undesirable experience for the user as they interrupt their daily activities. Furthermore, physiological biometric based recognition can potentially be used to authenticate an unconscious user [35].

A way to overcome such privacy and feasibility concerns is to utilize behavioral biometrics for user authentication. Behavioral biometrics measure the uniqueness in the behavioral patterns of users. Behavioral features include voice [58], keystroke gestures [18], motion gestures [19], gait or walking style [20], touch-gestures on a touch screen device [21] etc. Research show that human beings exhibit considerably unique behavioral gestures [19, 31].

Continuous authentication has been achieved by monitoring touch dynamics in smart devices [21]. Touch gestures have been modelled by recording the pressure, or force exerted during a touch gesture or by capturing the accelerometer readings during the touch gesture [57] etc. However, modelling touch gestures on wearables is difficult due to the small area of the touch interface. In fact, certain wearable devices like the Myo Armband does not have an embedded touch interface [4, 6]. Along the same lines, keystroke dynamics is not suitable for application to wearables, since most of these devices do not come pre-equipped with a complex keypad to model keystroke gestures well.

Wearable devices are best known for capturing activities or motion gestures. This natural feature of wearables can be utilized to uniquely identify a user. This method also eliminates the need of additional sensors since motion gesture recognition sensors are pre-embedded in wearables.

Negara et al. [31] in their paper show that motion gestures can help uniquely classify users even when they perform a simple basic activity like picking the phone to receive a call. Negara et al. [31] examined the uniqueness of arm movements for users with a 3-axis accelerometer. They achieved 87.8% accuracy when a user picks the phone from the table, and 90% accuracy when the phone is picked from the pocket.

A simple movement gesture would vary from person to person and is considered unique because of the difference in the physiological structure of the human body. The amount of force exerted by the different muscles to carry out a specific action may vary due to the variation in the muscular structure of the human body. The sensor readings would not be identical for the same reason [19, 31]. In the current literature, accelerometer [20], gyroscope [24], magnetometer [25] and GPS sensors [26] have been used to study user behavioral profile in this regard.

Accelerometer and gyroscope sensors are mainly used as motion sensors. Accelerometers record the acceleration of the body part they are attached to. They also detect device orientation. Whereas, gyroscopes record the rotational velocity of the body part they are attached to. Depending on placement, gyroscopes can detect altitude of the different body parts they are attached to. These sensors are each able to capture acceleration and rotational velocity in three dimensions. Sensor data from these together provide a 6-dimensional vector, which has complete motion information of the body part these sensors are attached to [27]. Popular smart devices come pre-equipped with these cheap, tiny, low-power sensors.

A magnetometer sensor on the other hand measures the strength of the magnetic field in three dimensions. Though, electrical activity of the human body does not produce noticeable changes in the strength of the magnetic field, the magnetometer sensor helps infer heading of the user wearing it. Similarly, GPS sensors help infer the location of the user wearing it. Magnetometer and GPS sensors may provide valuable information at a time of a theft attack if the attacker decides to change the location. However, if an illegitimate user happens to use the device while being at a location frequented by the legitimate user then location-based sensors may not sense an anomaly. In order to address this issue, in the current literature, location-based sensors have often been used in conjunction with other sensors such as the accelerometer sensor.

Gait pattern is a motion gesture based behavioral biometric. It has been extensively studied [20] in the current literature as a means of user identification and authentication and high accuracy has been achieved. Gait measures the uniqueness of the walk patterns as exhibited by a user. Nickel et al. [20] could achieve accuracy of 82% with a 3-axis accelerometer by monitoring motion gestures from gait patterns using KNN (K-nearest neighbors) classification algorithm. However, they utilize just a single sensor for authentication, which largely limits their performance.

Gadaleta et al. [32] achieved almost perfect accuracy by using both accelerometer and gyroscope sensors to measure gait patterns. They use convolutional neural networks and SVM. However, gait based authentication can authenticate users only when they are walking. Some physical effort is required as users need to walk for a few seconds to be accurately authenticated. A motion gesture based authentication approach, which can authenticate the user irrespective of the activity the user exhibits is shown in our approach. Such a system does not impose a constraint on what the user must do to be validated, ultimately leading to the development of a user-friendly system. This also does not interfere with the user's routine activities.

Yang et al. [24] could identify wearable users correctly using motion gestures from accelerometer and gyroscope sensor data. However, they asked users to perform a specific set of actions to train the model. During the verification phase, the users were asked to repeat the same actions. Similarly, SenSec (Zhu et al.) [28] achieved an accuracy of 75% in identifying users with accelerometer, gyroscope and magnetometer sensors. However, the users performed fixed pre-determined actions. During validation, the user was asked to repeat the specific gestures, based on which she would be authenticated. These methods are a good alternative to traditional authentication approaches. However, it would not authenticate the user continuously. These methods prompt the user to verify their legitimacy by following a script of actions in order to be authenticated. In addition, it requires the user to consciously participate in the authentication process, which might interfere with user's regular activities. Performing specific or determined gestures can be cumbersome to the user as well and a conscious user may never exhibit natural gestures. Such an authentication approach may be vulnerable to mimic attacks [24].

To overcome these issues, an authentication system should be built which would authenticate users unobtrusively by utilizing natural gestures, which users exhibit unconsciously, since natural gestures are difficult to spoof [19]. Lu et al. [32] present an unobtrusive method of user authentication. They used a combination of a supervised decision tree classifier and an unsupervised learning algorithm. It was demonstrated that even with 20% more data about 5% less accuracy was obtained when using unsupervised learning as compared to using supervised learning. Davidson et al. [33] in their study compared two algorithms: K-Nearest Neighbor (KNN) and Random Forests to authenticate smart watch users. They demonstrated that both KNN and random forests were robust to overfitting with KNN being robust against noisy data. However, they observed that KNN algorithm provides better performance with a high computational cost, with a requirement to tune k in each experiment. We see that with our approach better performance with lower complexity is obtained.

Lee et al. [25] achieved high accuracy of up to 95% by monitoring gestures using accelerometer, gyroscope and magnetometer sensors on a smartphone device. Chauhan et al. also presented a continuous authentication scheme for smart wearable devices [34]. However, they both use SVM, which provides high accuracy at the cost of quadratic computational complexity. Our approach yields high accuracy with gradient boosting on decision trees, which has a lower time complexity, and is thus computationally more efficient.

Few works have addressed the issue of adapting to a change in user gesture patterns. Behavioral features of a user may evolve over time. A well-trained model would fail to recognize a legitimate user eventually if the user gesture pattern evolves causing a change in the user behavioral profile. The classification accuracy would drastically reduce if user gesture evolution is not captured well. This calls for frequent or regular training of the data to build a new model to adapt to changes in user motion gestures. Lee et al. [25] in their paper talk about continuous rebuilding of the machine learning model to adapt to changes in user behavioral patterns. It is not very clear using how much or on what data the model would be retrained. Chauhan et al. [34] presented a method in which random samples from previous days' data are mixed with data from the present day daily as a method to adapt to changing user gesture patterns. Retraining a model regularly to capture user gesture pattern changes is extremely costly in terms of the acquired time and space complexity, especially when the size of the data is large and ever increasing. Furthermore, random sampling has disadvantages like over-representation or under-representation of a specific population of data. With random sampling, there are chances of losing important information from the data if resourceful samples are not picked.

We present a novel and an efficient approach to adapt to user behavioral evolution in this thesis. To reduce the computational cost of rebuilding the machine learning model, our approach updates the model by training it on top of the existing model. This way we do not retrain on data on which our model has already been trained in the past. We update our model by training it just on newer data to include information from the newer data. So, essentially, each time while updating the original model, the training happens just on newer data. With "newer data" we refer to the data, which has not been used for training in the past. This way we build an updated model, which has been trained on all the available data, but with lower computational cost. Since, we reuse the original model and then update it, we save on computational cost. Our approach thus yields high performance while saving time and computational resources.

CHAPTER 3

THE OVERALL APPROACH TO SECURE SMART WEARABLES

3.1 Threat Model of Our Approach

In our approach, we consider a situation in which an attacker has access to the following:

- Physical access to the smart wearable device
- Passcodes or PINs for unlocking the smart device
- Private data (e.g. email, text messages) available on the smart device
- Synced data that is available on the smart device as the smart wearable connects to other devices in an IoT environment.

An attacker may gain possession of private data stored on the smart device. Our aim is to generate a sensor data based behavioral biometric authentication system that would validate a user uninterruptedly. Thus, at the time of attack, the adversary would be forced to validate herself beyond the passcode validation step.

3.2 Assumptions

We make few reasonable assumptions in our approach, which are described in this section.

The presented approach uses accelerometer and gyroscope sensors from a wearable smart device to read user behavioral data. Depending on the collected data, a user profile is built during the training phase. The user profile in the form of the machine learning model is used to predict the validity of a user accessing the device continuously. Throughout the process, we heavily rely on the sensor data readings. The user validation step is dependent on the sensor readings as well. We assume that the sensors are not faulty or damaged. We also assume that the sensors used in our approach provide us reasonably accurate readings throughout the training and the validation steps. In addition, we assume that the sensor readings have not been tampered or modified by any means possibly by an attacker.

During an initial enrollment phase, the data is collected from the smart device for a period for building the machine learning model. Unless the first initial model has been built, the authentication process is inactive. We assume that during the initial training phase, the legitimate user uses the device and the data is safe from attacks. We also assume that an attacker does not have access to the generated model, and the model is secure from illegitimate modification. In summary, we assume that the components of the authentication software are secure from illegitimate access or modification. We assume that the authentication software is inaccessible to the device owner. Under such assumptions, an attacker as defined by our threat model would not have access to the authentication software, so our assumption is feasible in a real-world scenario.

It is worthy of mention that we primarily consider confirming a user against the smart device owner, since smart devices are generally personally owned devices which are not shared across multiple users.

We also assume that the user of the smart wearable device wears the device at the same part of the body throughout the training and the validation phases. It is common for users to wear the same smart watch on either of their wrists. The dominant and the recessive side of a human body may produce different sensor readings for the same user. Many smart watch manufacturers ask the users to specify if the user is wearing the smart watch in their dominant wrist or not for a specific usage session. To keep things simple, thus, for our study we assume that the user wears the smart device at the same body part throughout the training phase as well as the validation phase.

3.3 Our Overall Approach

Wearable devices come with an array of in-built sensors, the one common thing among these devices is their ability to monitor, and record user motion gestures with the help of these embedded sensors. These sensors along with their wide scale availability make wearables highly appropriate for implementing a motion gesture biometric based authentication system. An important advantage with wearables is that by their nature, these devices are always worn by user, thus capturing behavioral biometrics for authentication is easy and feasible. This in turn makes it possible to implement the continuous authentication scheme as well.

We present an approach to develop an authentication software for smart wearable users in this thesis. Our primary goal is to develop a user-centric continuous authentication software to secure smart wearables in the Internet of Things (IoT) environment by using user motion gesture based behavioral biometrics. Our approach has two major components:

1. The first component learns user behavior and generates a user profile in the form of a machine-learning model. This machine learning model is used to authenticate the user continuously throughout the usage session.
2. The second component updates the generated model periodically as required.

Our approach eliminates the need of conscious user inputs. We plan to hide the components of the authentication software from the user as part of abstraction. In addition, we plan to make the authentication software inaccessible to users to prevent unauthorized modification.

A smart wearable user would wear the smart device throughout the usage session. Thus, continuous stream of data is obtained which can be utilized for continuous authentication. Hence, the user can be authenticated throughout the usage session. This is exactly what we mean by "continuous authentication". Furthermore, smart wearables are designed to recognize motion gestures. We incorporate this natural feature of wearables in our approach. We utilize motion sensor data for building the user behavioral profile (machine learning model) which is also used to verify the legitimacy of the user.

As per our approach, we collect the motion gesture data of the user from the accelerometer and gyroscope sensors embedded in the wearable device. Additional features are extracted from the sensor data after which all the features are used as input for training the model. We implement scalable gradient boosting on decision trees to generate the machine learning model.

The machine learning model validates the user after receiving fresh sensor data during the validation phase. Please note however that the raw data must be passed to the feature extraction algorithm as input, the output from which is passed to the built machine-learning model as an input to output a decision. The decision of the model as an output tells if the user is legitimate or not. If the user's legitimacy is verified as part of the decision from the model output, then the validation service continues to run in the background. If a user is determined to be an illegitimate user, then the authentication service locks the device and the user is required to validate herself.

As part of adaptation, the previously generated model is updated in the event of a substantial dip in the accuracy of the decision-making process. It is responsible for updating the model with freshly generated data by training it on top of the existing model. The process of updating the model happens in the *model generation (training the tree model)* step. However, it needs to be initiated by the adaptation process.

Popular wearable devices store user data for a period, after a decided period, a fresh new model can be generated by replacing the existing model. This is because older data may not be as useful after a certain point of time. Replacing the model after a substantial period (e.g. few months) is a more feasible option than replacing the model frequently instead of updating the model. This is because training models is an extremely costly event in terms of the acquired computational time and resources. Figure 6 shows our overall approach, which can be summarized as follows:

(Step 1) ***Data Collection:*** User motion gesture data is collected in this step.

(Step 2) ***Feature Extraction:*** The feature extraction algorithm gets the collected data as input and statistical features are extracted as an output at the end of the step.

(Step 3) ***Training the Gradient Boosted Tree Model:*** The features are passed on as input to the scalable and optimized Gradient Boosted Decision Tree (GBDT) algorithm to train the model on the input features in this step. The learning model is generated and saved at the end of this step.

(Step 4) ***Authentication:*** The user is validated in this step. The user validation service runs in the background for authentication purposes.

(Step 5) ***Adaptation:*** The gradient boosted tree model that is built as part of *step 3* is updated in this step as part of adaptation. This step is to an extent dependent on the feedback received from *Step 4*.

During the initial training phase, the sensor data is collected and the initial machine-learning model is generated. After the model is built and saved, the continuous authentication service is activated. The authentication service receives fresh data from the output of the feature extraction algorithm. The features are passed on as input to the saved learning model and the result determines validity of the user.

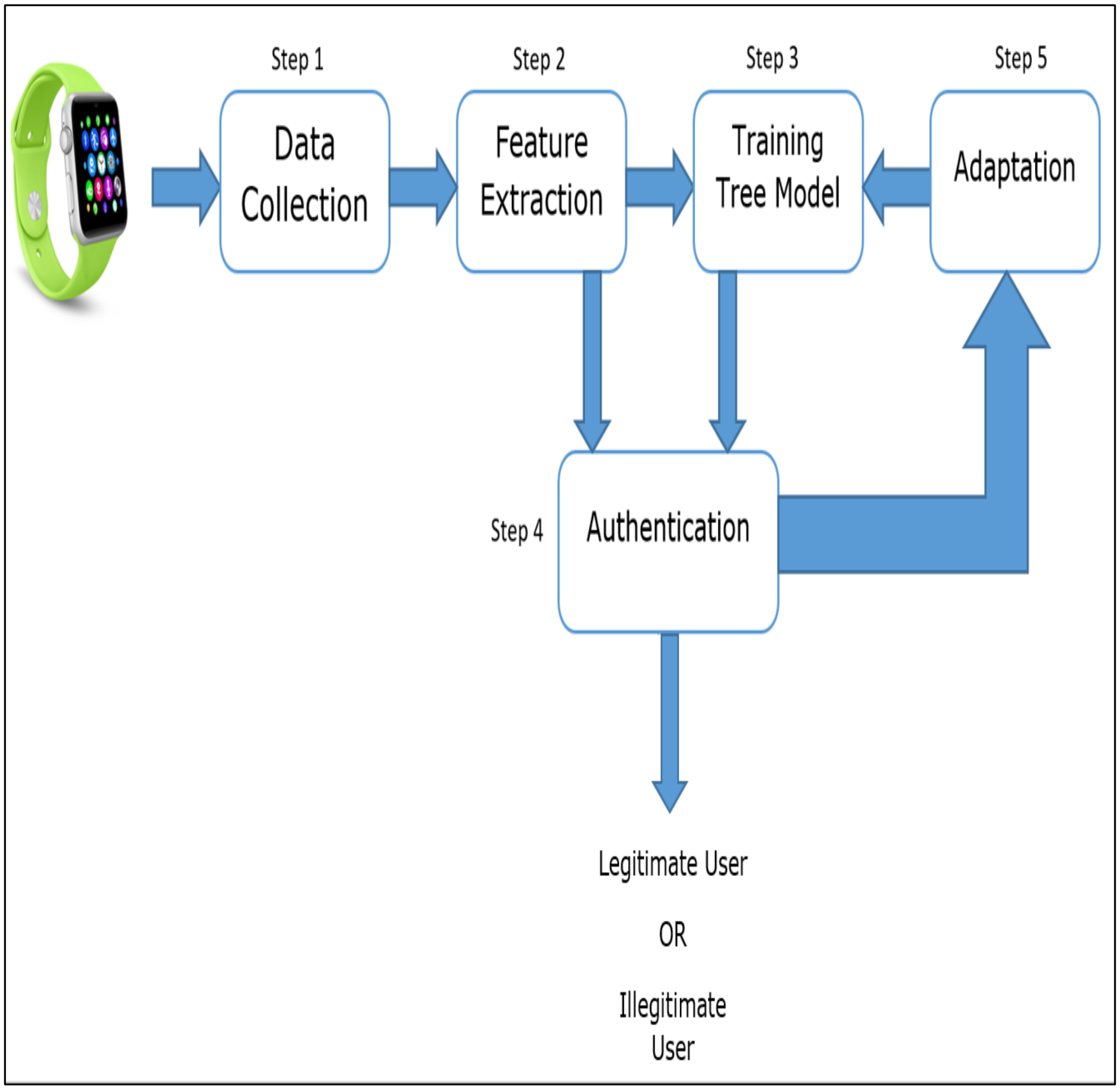


Fig 6: Our Approach

The authentication service sends feedback to the adaptation service regarding model performance. The adaptation service decides to update the model if the accuracy dips substantially. It interacts with the model generation service to update the existing model. The updated model is saved in place of the original model. The steps are repeated henceforth.

In the upcoming chapters, further details of the steps of our approach to develop the authentication software are provided.

CHAPTER 4

DATA COLLECTION

This chapter discusses the very first step of our approach. It is an important step because all the remaining steps or components of our approach depend on it.

In the data collection step, readings from accelerometer and gyroscope sensors are collected. Accelerometer sensors are used to record the acceleration and gyroscope sensors are used to record the rotational velocity of the body part they are attached to. There are few advantages of using these sensors. Firstly, these sensors are each able to capture acceleration and rotation in three dimensions and together they can provide complete motion information of the body part they are attached to [27] in six dimensions. This is a major reason as to why in our study we use both the accelerometer and gyroscope sensors. Secondly, popular smart devices come pre-equipped with these cheap, tiny, low-power sensors. Table 1 provides a list of smart devices with the accelerometer and the gyroscope sensors. Last but not the least, these sensors do not need user permission to work. Thus, they are able to collect user data without the need of active user intervention. Figure 7 shows the embedded sensors of the Apple Watch.

A wearable device user wears the device during the usage session. Thus, data can be continuously collected during the usage session. Initially, data is collected for training the gradient boosted tree model. Later, data is collected primarily for the purpose of authentication of the legitimate user once the trained tree model has been generated. Collected data from this step is also used to initiate the process of adaptation. Data collected from this step is always passed to *step 2* for feature extraction.

Table 1: Popular Smart Devices with Accelerometer and Gyroscope Sensors

Smart Device	**Accelerometer**	**Gyroscope**	**Device Type**
Apple Watch Series	✓	✓	**Smart Watch Or Fitness Bands**
Fitbit Surge	✓	✓	
Alcatel One Touch Watch	✓	✓	
Motorola Moto 360 Watch	✓	✓	
LG G Watch	✓	✓	
Asus Zen Watch	✓	✓	
Huawei Watch	✓	✓	
Samsung Gear Watch	✓	✓	
Samsung Gear Watch	✓	✓	
iPhone 6	✓	✓	**Smart Phone**
Nexus 7	✓	✓	
Huawei P8	✓	✓	
Samsung S6	✓	✓	
Myo Armband	✓	✓	**Gesture Control Arm Band**
Nymi	✓	✓	**Heartbeat Monitor**

Fig 7: Embedded Sensors of the Apple Watch

At the end of the data collection step, accelerometer and gyroscope readings in the three axes or the three dimensions are obtained. Table 2 shows the data that is obtained as part of data collection.

Table 2: Data Collection

Acceleration in x-axis	**Accelerometer sensor**
Acceleration in y-axis	
Acceleration in z-axis	
Rotational velocity in x-axis	**Gyroscope sensor**
Rotational velocity in y-axis	
Rotational velocity in z-axis	

CHAPTER 5

FEATURE EXTRACTION

The collected data from *step 1* of our approach is passed to the feature extraction algorithm as input for feature extraction. In this chapter, the feature extraction algorithm is discussed in detail.

The goal of feature extraction is to derive values or features from the original data to discover meaningful insight from data; reduce noise, and redundancy. Feature extraction is a useful step that helps estimate the classification parameters of the supervised machine-learning model more accurately. We primarily derive statistical features from the raw sensor data. Table 3 provides a list of all the features extracted in this step.

Firstly, we extract the magnitude of acceleration from the accelerometer readings in three dimensions. Similarly, we calculate the magnitude of the rotational velocity from the gyroscope readings in three dimensions. Next, we begin the process of extracting statistical features. To prepare the dataset for feature extraction, we divide the entire dataset into non-overlapping frames or windows. Each window consists of fixed sized data points. Segmentation of sensor data into uniform windows has been found to be effective in the current state of the art [49] as a data preparation step prior to feature extraction. The number of data points per window can be determined by considering the sampling rate of the sensors. For a sampling rate of 30-60Hz (the general sampling rate for accelerometer and gyroscope sensors for majority of smart devices), a 10s window (rounded off) would have around 300 data points. Considering the information, we divide the dataset in a way such that each non-overlapping window has 300 data points. From the data points of each bucket for both the sensors per axis, and for the magnitude values per sensor, we extract statistical features. Some of the statistical features are the result of a mathematical aggregate function (e.g. mean, maximum, minimum etc.) which are computed over each data point per window. The result from the aggregate is replicated across all the data points of the window. Algorithm 1 is the feature extraction algorithm for our approach.

<u>Algorithm 1: Feature Extraction</u>

Note: All the calculated values are saved under the corresponding features.

```
1. Calculate magnitude of accelerometer readings.
2. Calculate magnitude of gyroscope readings.
3. Split the data set into $B$ frames of fixed window size $k$
4. For B = 1 to n
   4.1. Calculate mean, standard deviation, variation, maximum, minimum, skewness,
        kurtosis, root mean square (RMS) for all $k$ samples.
   4.2. Repeat 4.1 for accelerometer readings for accelerometer readings for 3 axes,
        gyroscope readings for 3 axes, and magnitude values for accelerometer and
        gyroscope.
   4.3. Replicate each calculated value across $k$ samples
   4.4. End For.
5. Calculate waveform length ($x_{i+1}$-$x_i$) for accelerometer readings for 3 axes.
6. Calculate waveform length ($x_{i+1}$-$x_i$) for gyroscope readings for 3 axes.
7. Calculate average absolute difference of mean for accelerometer readings for 3
   axes, gyroscope readings for 3 axes.
8. For B = 1 to n
```

```
    8.1. Calculate range, first quartile, second quartile, third quartile
    8.2. Repeat 10.2 for accelerometer readings for 3 axes, gyroscope readings for 3
         axes
    8.3. Replicate each calculated value across *k* samples
    8.4. End For.
9.  For B = 1 to n
    9.1. Calculate binned distribution values with bin size = 10.
    9.2.        Store each bin result as a separate column of feature. We should have
                10 bin columns. Replicate each bin's value across k samples.
    9.3.        Repeat 9.1 and 9.2 for accelerometer readings for 3 axes, gyroscope
                readings for 3 axes.
    9.4.        End For.
10. End.
```

After data preparation, additional features are extracted from the data. The features obtained from the feature extraction algorithm as output, are passed as input to train the model. The features are input to the generated model after training for continuous authentication purposes. Data preparation and feature extraction has a key role in model performance and must be carefully executed.

Table 3: Extracted Features

Feature	Aggregate Function	Extracted From
Magnitude	✗	**3-axis accelerometer + 3-axis gyroscope**
Mean	✓	**3-axis accelerometer + 3-axis gyroscope + Magnitude from 3-axis accelerometer + Magnitude from 3-axis gyroscope**
Standard deviation	✓	
Variation	✓	
Maximum	✓	
Minimum	✓	
Skewness	✓	
Kurtosis	✓	
RMS	✓	
Waveform length	✗	**3-axis accelerometer + 3-axis gyroscope**
Absolute difference of value from mean	✗	
Average absolute difference from mean	✓	
Binned distribution (10 bins)	✓	
Range	✓	
First quartile	✓	
Second quartile	✓	
Third quartile	✓	

CHAPTER 6

TRAINING THE GRADIENT BOOSTED TREE MODEL

In this chapter, we discuss in details the core of our approach: training the gradient boosted decision tree model. This is *step 3* of our approach. We start by providing a brief background of the classification problem at hand. We then discuss in greater depth the machine learning model generation algorithm.

Information from the data collection step is passed on as input to the feature extraction algorithm for processing. The features obtained as output are fed to the tree building algorithm for generating the tree model. The machine-learning model is prepared in this model generation step. The generated model learns the user behavior based on the data collected from the accelerometer and the gyroscope sensors, and the features extracted. The model is saved and made available to the authentication service, about which we would discuss in the upcoming sections. The generated model at the end of the step is used to determine the legitimacy of the user as part of the process of continuous authentication.

6.1 Background

Machine learning algorithms provide an ability to predict on new data on being trained upon a learning dataset. Mathematically, a machine learning model refers to a mathematical function, which predicts the target variable y_i as the output, given the input x_i, where *i* refers to an instance of data. A supervised machine learning algorithm is one, which uses a training dataset with known or labelled *target* values to learn the features. The *target* variable is the feature on which a prediction must be made. To generate a machine learning model, the model is trained on a training dataset and then it is evaluated or tested against a test dataset. A good machine learning model should not only predict well on training data, but also predict well on test data. The evaluation metrics determine the quality of the generated model.

The goal of the machine-learning model of our approach is to predict if a user is the legitimate user. The target values of our learning model are "authentic" and "not-authentic" for the legitimate user and the illegitimate user respectively. This is a 2-class supervised classification problem where each instance of the training dataset is associated with either of the two pre-determined class labels (target values). The pre-determined labels are used for training or learning the user features. We implement scalable and regularized gradient boosting on decision trees to learn the user features. In the later sections of the chapter, we would describe the training algorithm and the evaluation metrics for testing the generated machine learning model.

6.2 Decision Trees

Decision trees are used as the base classifier for building the model. The objective of a decision tree classifier is to predict the target variable by learning classification rules construed from the features of the input data.

Decision tree classifiers have several advantages and the reasons as to why we use decision trees as the base classifier are discussed as follows: Decision trees are powerful classifiers who can model non-linear relationships between features and the target variable. The decision tree model is good at handling data with numerical features and features coming from different sources without much tuning. This is very much applicable to our problem at hand since we get data from multiple sources - accelerometer and gyroscope sensors each of which records data in 3-dimensions. Furthermore, trees require very little data preparation or normalization because they can handle qualitative predictors. Last but not the least, decision trees are known for being simple and intuitive for analysis and explanation purposes.

A decision tree comprises of one *root node*, one or more general *nodes*, *leaf nodes*, and *edges* coming out of each node. Each *node* branches out into several *child nodes* via *edges.* Each node represents an *attribute* or *feature*. The appearance of a feature in a tree tells us about the significance of the related attribute in forming the decision tree model. The *leaf nodes* represent class labels. The *edges* coming out of each node represent the possible values of the feature node from which they initiate. All the features are passed in as an input to the decision tree algorithm. The output is the generated decision tree model. Figure 8 represents the decision tree structure.

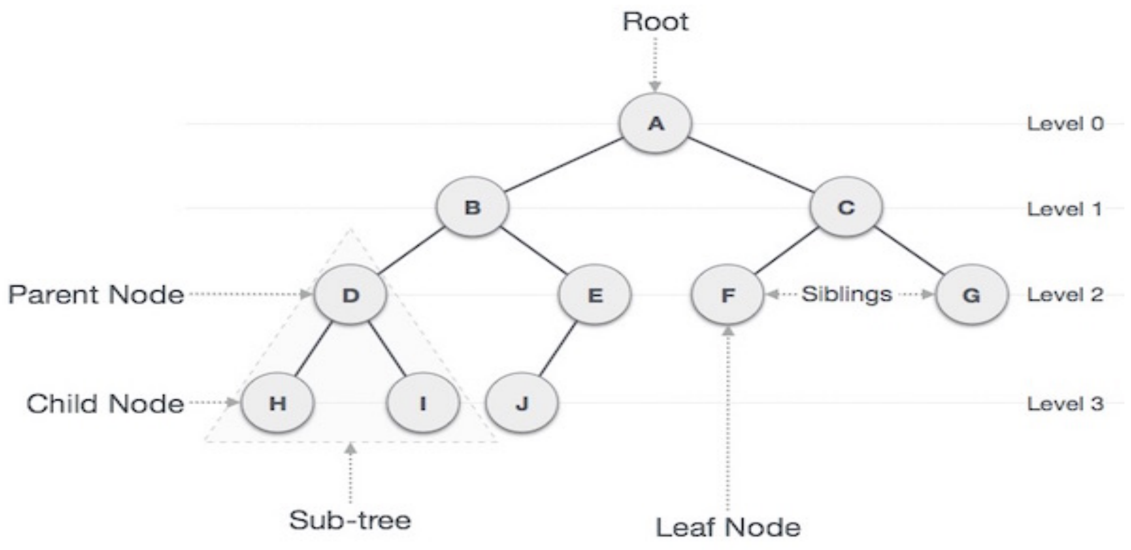


Fig 8: Decision Tree Structure

6.3 Feature Selection

An important characteristic of decision trees is automatic feature selection as part of classification. Feature selection is a process of selecting a relevant subset of features from the set of all the available features. Relevant features can be defined as features which are neither irrelevant nor redundant to predict the target variable in the machine learning problem at hand. An irrelevant feature on the other hand does not affect the prediction at all and a redundant feature does not add value to predicting the target. Reducing the number of features is also called dimensionality reduction. Moreover, effective feature selection leads to less memory utilization [39], reduced train and test time [40]. Feature selection also reduces chances of overfitting [41].

Feature selection surely helps increase model performance. However, an additional feature selection step may be expensive to implement in terms of computational time and resources. Decision trees provide a natural way of optimization by combining the feature selection and classification steps. We utilize these advantages of trees in our approach, which eliminate the need of having an additional feature selection step.

The standard decision tree model follows the c4.5 algorithm [54]. We implement this standard algorithm in our approach. Decision trees work by splitting on attributes or features at each node. An attribute is selected for a split based on some estimation criteria. The estimation criteria select the best feature to split on. Information gain and entropy reduction provide a measure of the estimation criteria [42]. The criteria try to discover the feature that best discriminates the target class values.

Entropy reduction is an estimation criterion, which is used to perform feature selection as part of building the decision tree. Entropy measures the randomness of instances of data. In other words, it measures the homogeneity of data in a sample space. Mathematically, entropy for a binary class can be given by equation (1), where *P(a)* stands for the probability of occurrence of class value *a* and *P(b)* stands for the probability of occurrence of class value *b*:

$$\boxed{\text{Entropy} = - P(a)*\log_2(P(a)) - P(b)*\log_2(P(b))} \qquad (1)$$

Thus, from equation (1), entropy of the data with 50% of each class is 1 (maximum entropy) and for a distribution in which either of the class labels are present, the entropy is 0. A training set with maximum entropy is good for learning since the data distribution is *unbiased*. In the upcoming sections, we would talk more about bias and variance.

Information gain on the other hand measures the importance of a given attribute/feature. In other words, it measures how an attribute helps discriminate the training set based on the target variable. It selects the best feature to split on at a given time. Its value is calculated from the difference between the parent node entropy and the expected weighted average of the entropy of children nodes that would be formed if a split on the attribute occurs. Mathematically, information gain can be represented by equation (2), where the first term represents the entropy before a split occurs and the second term represents the expected entropy if a split occurs. Equation (2) provides the information gain (*S, A*) of a feature or attribute *A* with respect to sample space *S* [42]. In equation (2) the term *values(A)* represents all possible values of feature *A* with respect to *S*, S_v is the subset of *S* for which *v* is a feature value of attribute *A. The* term $Entropy\ (S_v)$ is the sum of entropies for each subset *v.*

$\sum_{v \in values(A)} |S_v|/|S|(Entropy\,(S_v))$ term thus calculates the weighted average of the entropy of subset *v*.

$$\boxed{\text{Information Gain (S, A) = Entropy(S) - } \sum_{v \in values(A)} |S_v|/|S|(\text{Entropy }(S_v))} \qquad (2)$$

For each node of the tree, the information gain is calculated for each feature, and the feature with the maximum information gain is chosen for the split. An expected reduction in entropy is caused due to a split.

Decision trees thus work by splitting the sample set into subsamples based on the values of the attributes. The process of splitting nodes is done recursively via recursive partitioning. This is a greedy method of problem solving. The C4.5 algorithm for generating a decision tree model is provided by Algorithm 2.

<u>Algorithm 2: Decision Tree algorithm</u>

```
1. Check if all instances belong to same class.
    1.1 If yes, go to step 1.2. If no, go to step 2.
    1.2 Create a leaf node with that class.
2. Check if none of the features provides an information gain.
    2.1 If yes, go to step 2.2. If no, go to step 3.
    2.2 Use the expected value of the class to create a decision node higher
        up the tree.
3. Check if instance of a new class label appears.
```

    3.1 If yes, go to step 2.2. If no, go to step 3.

    3.2 Use the expected value of the class to create a decision node higher up the tree.

4. For each attribute A_i, calculate the information gain if a split of A_i occurs.
5. Let $\widehat{A_\iota}$ be the attribute, which provides maximum information gain if a split occurs.
6. Create a classification *node* that splits the attribute $\widehat{A_\iota}$.
7. Repeat all steps for subsets obtained by splitting $\widehat{A_\iota}$. Add new *nodes* as children of current *node*.
8. End.

Decision trees are simple, easy to understand, and interpret, and have several other advantages. But, a major limitation of decision trees is that they easily *overfit.* Because of which decision tree learning does not generally achieve high prediction performance, when compared to other methods. This is because a small change in the data may produce a very different tree leading to a large difference in prediction. Tree learners often create very complex trees, which lead to overfitting. Overfitting occurs when machine learning models do not generalize well beyond the training set. Decision tree models are very sensitive to small changes in data and do not perform well on new data. Boosting helps resolve such limitations of decision trees.

6.4 Gradient Boosting

Boosting [44] is an ensemble method of learning which combines multiple weak learners to generate a strong learner. Strong learners are learning algorithms that produce classification results, which are close to true classification. In other words, strong learners yield high performance. Weak learners on the other hand do not provide high performance and perform little better than random guessing. Shallow decision trees or decision trees with little depth (levels) are generally considered as weak learners since they do not fit the train data very well. Boosting is a general technique which is known for producing high performance results by combining weak or moderately weak learners to develop a strong learner.

Boosting optimizes the overall model building process by reducing bias and variance. Bias is caused when an algorithm underfits the data. Bias occurs due to wrong assumptions in the learning process, which can cause the training algorithm to miss important relationships between the features and the target variable. While variance is an error that occurs if the model is sensitive to small changes in the data. Variance leads to overfitting and happens when the trained model does not generalize well beyond the training set.

In our approach, we use boosting with decision trees as the base learner. It uses multiple weak or shallow decision trees to build a model, which generalizes well. The boosted decision tree algorithm combines the predictions of multiple trees to predict a class value. It combines the prediction from the several trees in an additive manner. Our approach has the advantage of the underlying tree structure, and with boosting we get a more effective algorithm for learning the final nonlinear resulting ensemble of trees. Equation (3) provides the boosted decision tree ensemble equation where $f_i(x)$ stands for a decision tree and $g(x)$ stands for the boosted tree model.

$$\boxed{g(x) = f_0(x) + f_1(x) + f_2(x) + \cdots f_n(x)} \quad (3)$$

The gradient boosting [43] algorithm is a specific boosting technique. It combines base learners, which are decision trees in our case with a method called gradient boosting. The name gradient boosting is derived from the optimization function of gradient descent, since gradient boosting uses the gradient descent algorithm for boosting. The gradient descent method is used to minimize the loss function when adding trees. Conceptually, a loss function is some function of the difference between the estimated and the true values of the labels of the instances. Commonly used loss functions: mean squared error and logistic log functions are given by equation (4) and equation (5) respectively. The following equations are formulated considering that in a training data sample of known values of x_i and target class values of y_i, where *i* represents each data point and *n* is the total number of data points, the training set is $\{x_i, y_i\}$.

$$\boxed{L\,(y_i, g\,(x_i)) = \frac{1}{N}\sum(y_i - \hat{y}_i)^2} \quad (4)$$

$$\boxed{L\,(y_i, g\,(x_i)) = \frac{1}{\log 2}\log\,(1 + e^{-yf(x)})} \quad (5)$$

Logistic log function being continuous helps utilize gradient descent and is a common classification loss function. In our approach, we use the logistic log function, which we call log loss. It is used to determine model accuracy and evaluate the model.

Equation (6) shows the gradient descent optimization function where *t* stands for the number of iterations and η stands for the step size. If we want to optimize differentiable function *f(x),* the gradient descent algorithm works by iteratively finding:

$$\boxed{x_{t+1} = x_t - \eta \frac{\partial f}{\partial x}\Big|_{x=x_t}} \qquad (6)$$

Let *g(x)* be our model function. From equation (3), we can express *g(x)* for each iteration as equation (7), where symbols have their usual significance:

$$\boxed{g_t(x) = \textstyle\sum_0^{t-1} f_i(x)} \qquad (7)$$

Let $g_t(x)$ be the trained classifier at iteration *t,* and $L(y_i, g(x_i))$ be the loss function of the training set $\{x_i, y_i\}$, then at each iteration (*t+1*), g_t leaps towards negative gradient descent by $\frac{\partial f}{\partial x}$ amount with step size of η. Here, f_t is chosen to be the argument of the minimum, i.e. the value of x is chosen such that the minimum value of *f(x)* is obtained. Equation (8) calculates value of f_t.

$$\boxed{f_t = \ arg\ min_f \ \textstyle\sum_{i=1}^{N} \left[\frac{dL(y_i, g(x_i))}{dg(x_i)}\Big|_{g=\ g_t} - f(x_i)\right]^2} \qquad (8)$$

The gradient boosted algorithm combines base learners, which are decision trees in our case with a method called gradient boosting using the gradient descent optimization method. At iteration *t+1* the gradient boosting algorithm computes g_{t+1} as provided by equation (9):

$$\boxed{g_{t+1} = g_t + \eta f_t} \qquad (9)$$

The greedy gradient boosted algorithm [43] as proposed by Friedman is implemented in our approach. It can be implemented with any differentiable loss function. However, since we have a classification problem, we use the logistic log loss function, which is common to use [43]. The algorithm aims to minimize the average value of the specific loss function while learning. Let $\hat{y}$ be the optimized prediction value. Then the algorithm tries to predict the best value of $\hat{y}$ for *f(x).* This is given by equation (7). We would describe the algorithm of the gradient boosted model next. We learn one tree at a time in the gradient boosted algorithm. Let us consider the training set to be $\{x_i, y_i\}$, *t* to be the number of iterations, f_i be the tree models, $\hat{y}$ is the prediction value at each iteration and *k* be a constant, then we get the following equations at each iteration:

t = 0 $\hat{y}_0$ = k

t = 1 $\hat{y}_1 = \hat{y}_0 + \mathrm{f}_1(x_i)$

t = 2 $\hat{y}_2 = \hat{y}_1 + \mathrm{f}_2(x_i)$

…

At t^{th} iteration $$\hat{y}_t = \hat{y}_{t-1} + \mathrm{f}_\mathrm{t}(x_i) = \sum_{m=1}^{t} \mathrm{f}_\mathrm{m}(x_i) \qquad (10)$$

Thus, the algorithm of gradient boosting tries to fit a new decision tree to the residual at each iteration. At this point, let us consider *g(x)* is the prediction function that we would learn to minimize the loss function. We aim to find $\widehat{g(x)}$, which is the optimized function for our problem. We aim to find a suitable *g(x)* from a class of non-linear function *G* to minimize the loss function. In our case, we consider $\bar{x}$ to be the input vector such that *x = {x [1], x [2], … , x[d]}* and a set of inputs x_i = $\{\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_\mathrm{n}\}$ such that *X € S* where *S* is the training set, and *d* is the total number of dimensions, and *Y* is the target vector such that y_i = $\{\mathrm{y}_1, \mathrm{y}_2\}$, as we have a binary class problem. A binary class has two labels. Equation (11) for our case defines $\widehat{g(x)}$. $L\big(y_i,\ g(x_i)\big)$is the logistic log loss function.

$$\boxed{\widehat{\mathrm{g(x)}} = \arg\ \min_{\mathrm{g} \in \mathrm{G}} \mathrm{L}\big(\mathrm{y_i},\ \mathrm{g(x_i)}\big)} \qquad (11)$$

From equation (10), we can say that $\hat{y}$ can be represented as a function, which must optimize the coefficients of the parameters of the input function $f_i(x)$. Thus, from equations (11) and (10) we can say that $\widehat{g(x)}$ depends on two parameters: coefficients of *t* decision trees and each decision tree rule $f_i(x)$. Therefore, there are c_t coefficients that can be optimized and each $f_i(x)$ is a base learner, which can be optimized. So, our overall algorithm must optimize all coefficients of the base learners. Consider the base learners to be trees with *M* terminal nodes and each base learner before fitting to be *h*. This is done with an update step that is used to optimize the coefficients. Algorithm (3) is the gradient boosted decision tree algorithm [43].

Algorithm 3: Gradient Boosted Decision Trees

```
1. $g_0(x) \leftarrow arg\ min_g\ L(y_i,\ g(x_i))$        //Initialize model with constant value
2. For t = 1 to T
    2.1. $\hat{y}_t \leftarrow \frac{dL(y_i, g(x_i))}{dg(x_i)}|_{g=g_t}$
    2.2. Fit a tree $P$ with leaf nodes $\{ h_{t,m} \}_{m=1}^{M}$
    2.3. For m = 1 to M
        2.3.1. $B_{t,m} \leftarrow arg\,min_{B \in S}\ L(\hat{y}_t, g_{t-1}(x_i) + B \,.\, h_{t,m}(x_i), y_i)$
    2.4. $g_t(x) \leftarrow g_{t-1}(x) + \eta \sum_{m=1}^{M} B_{t,m} .\, h_{t,m}(x_i)$
    2.5. End For
3. Return g(x) = $g_t(x)$
4. End.
```

Our approach is based on the above algorithm. However, an explicit regularization scheme has not been proposed in [43]. We implement regularization to reduce complexity of the model. Regularization helps prevent overfitting by reducing the complexity and is an important optimization step, which helps in generalization beyond the training dataset.

6.5 Regularization

Regularization helps prevent overfitting by reducing complexity of the model. Pruning is a regularization technique. Pruning is a method of reducing the complexity of decision trees by eliminating sections of trees which do not help classify instances well. Thus, pruning helps prevent overfitting. The following additional checks were added to the decision tree algorithm (Algorithm 2) for pruning the decision trees. Please note that the tree building algorithm step of Algorithm 3 also incorporates the pruning steps as a means of optimization as follows:

1. Check if minimum child weight = c
   1. If min (child weight) < c: skip split; continue to next corresponding step of algorithm;
   2. else: continue to next step of algorithm

Explanation: In our approach, c = 5. Minimum child weight refers to the minimum number of observations required at a leaf node. More formally, it refers to the required minimum sum of instance weights at each node. Larger values of minimum child weight yield simpler trees, which generalize well beyond the training data. If the tree building step results in a leaf node with the sum of instance weights less than the provided minimum child weight value of 5, then the algorithm will not split that node. This is a pruning technique. Pruning can be defined as a method of reducing overfitting in decision trees.

2. Check if minimum loss reduction value = L
   1. If min (loss function) < L: skip split; continue to next corresponding step of algorithm;
   2. else: continue to next corresponding step of algorithm

Explanation: In our approach, L = 7. The minimum loss reduction value determines if a split of the node/farther partition of the node would be carried out. If the loss function is not reduced sufficiently by at least 7 units then a split is not carried out. This is also a pruning technique. Here the expected loss reduction is considered, i.e. the loss function value if the split were to occur is considered, if it is less than 7, then the split does not occur.

3. Break further partition if maximum height of tree = D.

Explanation: In our approach, D = 4. The height = 4 units can be considered a reasonably good height (or level) for trees. This is because deeper trees increase complexity of trees which lead to a tighter fit increasing the chances of overfitting. The tree height lower than D on the other hand might not fit the data very well based on our approach.

We implement the following regularization measures to the gradient boosting algorithm (Algorithm 3):

1. The maximum number of iterations (t) is restricted to 20
2. The step size(η) is kept equals to 0.3.

Explanation: At each iteration (t), a new tree is created. Usually more the number of iterations, better the accuracy. With fewer iterations however, model may not fit the data well. With too many iterations, a tighter fit to the training data is obtained. Such a tight fit may lead to overfitting. In addition, the training and prediction time scales linearly with additional trees. Thus, the number of iterations is restricted to 20 based on our approach, since controlling the number of iterations reduces overfitting. Since, the step size or the shrinkage parameter works in conjunction with the number of iterations, care must be taken to fix the step size according to the number of iterations and vice versa. Shrinkage parameter of less than 1 has been found to be beneficial as an optimization step in the gradient descent algorithm [35]. Thus, we keep the step size to a value which is less than 1. A smaller step size than the one we use in our approach may take forever to converge, or would take too many iterations to converge. Whereas, a larger step size may not converge or may converge at infinity.

So far, we discussed the algorithm to train the model. We would train the model on a training dataset to build the machine learning model. The built model is tested or evaluated on a testing dataset next. The built model gets deployed once the evaluation results are satisfactory. The deployed machine learning model drives the continuous authentication scheme. In the upcoming sections, we talk about the data of the training set and test set, and the evaluation metrics.

6.6 Reducing Bias

Bias in statistics can be defined as the systematic preference that is present in the data of the sample set, which results in erroneous and misleading data analysis based results. A machine-learning algorithm might miss important relationships between the target variable and the features in the presence of high bias. As discussed earlier, it is thus important to reduce or eliminate bias in the training set before we start to train the machine-learning model. This also helps us with benchmarking analysis. If we have a binary classifier, random guessing would yield 50% accuracy. This is because probability of correctly predicting a class label is 0.5 in case of a binary classifier. A machine-learning model should at least provide accuracy better than the benchmark value of 50%. The above statements hold true for an unbiased training sample set. Thus, in order for us to execute benchmarking analysis, bias needs to be eliminated or minimized from the training set. An inherent error of bias results if the sample space does not have enough representative data of population of a class label. To reduce bias, we take approximately equal samples of the *"authentic"* label and the *"not-authentic"* label for the legitimate and the illegitimate user respectively. This helps us prevent a problem of underfitting and helps us evaluate the model in the true sense. We would dive deeper into further discussion on how the data for each of the class labels gets selected in the upcoming chapters.

6.7 Cross Validation

We train the model on a training set and test the model performance for evaluating the model on a separate test set. This process is done by cross validation. Cross validation helps avoid overlap between training and test sets for better evaluation. What we have realized from the discussion so far is that the base learners learn from a subset of data and their predictions are combined to predict the target value. It is important to shuffle the data set before we begin the process of training or before cross validation. This is because the subset on which the base learner is being trained may not have sufficient information for all the class labels – with a high bias. Shuffling is an important model evaluation step when data is coming from multiple sources [51]. We ensure that the subsets have minimum bias to avoid underfitting of data.

The parametric values are determined, and tested with leave-one-out k-fold cross validation. We implement 10-fold cross validation (with k = 10) in our approach. Leave one out k-fold cross validation is a method in which the dataset is divided into *k* samples, out of which (*k-1*) samples are used as the training set and remaining one set if used as the testing set. This process is repeated k times with each of the k subsamples used only once as a testing set. The K results are then averaged to estimate the performance of the model. This method helps check for overfitting or allows us to estimate how the model would generalize to an independent dataset.

During training 5% of the data is used as the validation set which is separate from the test set. The training and the validation sets are both used during training of the machine-learning model. The primary feature of a validation set is to see how well the model performs on the training set. In other words, the validation set helps provide an estimation of how well the model has been trained. In addition, separate training and validation sets help control overfitting. For instance, if the accuracy of the model increases on the training set and not on the validation set, it could be a sign of overfitting. One should stop further training at that point. On the other hand, the performance of the model on the test set tells us about the quality of the model. It gives us an idea as to how the model would perform upon deployment.

6.8 Tree Model Evaluation

To evaluate the model, several evaluation metrics were considered. The following metrics were used to evaluate our model:

- *Accuracy*: It provides the percentage of correct predictions for the given target value.
- *Confusion Matrix*: It is presented in a table format, which lists the number of true positives, false positives, true negatives and false negatives.
- *EER*: EER stands for equal error rate. The point of the ROC curve at which the false positive rate and the false negative rate are equal is given by the EER value. The value is commonly expressed as a percentage.
- *False Acceptance Rate*: It calculates the ratio of the number of false positives thrown to the total number of predictions made.
- *False Rejection Rate*: It calculates the ratio of the number of false negatives thrown to the total number of predictions made.
- *Precision*: Precision is a measure of relevancy. It is calculated as the ratio of the number of true positives to the sum of true positives and false positives.
- *Recall*: Recall's value provides a measure of the number of truly relevant results returned. It can be defined to be as the ratio of the number of true positives to the sum of true positives and false negatives.
- *f-1 score:* It calculates the harmonic mean of precision and recall.
- *Log loss*: It computes the logarithmic loss function for the predicted target and the actual target values. The log loss function formula has been discussed in the earlier sections.
- *ROC curve*: ROC curve stands for Receiver Operating characteristics (ROC). It is a graphical curve, which plots the true positive rate vs the false positive rate at various threshold settings.

- *AUC*: AUC stands for the area under curve of the ROC curve. It computes the probability that the binary classifier will rank a randomly chosen positive example higher than a randomly chosen negative example. A higher value of AUC indicates a good classifier.

The discussed metrics help evaluate the built model on the test set. A good model would perform well not only during training but also during testing. A model with satisfactory evaluation results is selected for deployment. The deployed model drives the continuous authentication method.

With our approach, overfitting can be prevented. So, the built model can generalize well and hence is more scalable. The gradient boosted decision tree scales in O(nlogn) time during training, where n represents the number of instances used for training. This includes the feature selection step. It takes constant testing time in terms of time complexity, which means that upon deployment, the model would predict in constant time or in O(1) time in terms of time complexity. This is reasonably better than existing algorithms, which are much more complex in terms of time complexity. Because of which, the training algorithm scales well with our approach. Also, the time complexity of this algorithm can be further optimized using approximation algorithms.

Furthermore, as per our approach, the machine-learning algorithm can also be trained using distributed computing frameworks [59]. A distributed computing framework involves a collection of independent computers (also called as nodes or clusters) which are interconnected via a network and can work in collaboration on a computational task. Each node or cluster can work on a portion of the task to achieve a computational result faster than with a single computer. The individual computers are cheaper and have lower processing power than a single powerful computer in general. Distributed computing especially helps when the dataset is very large and training on a single machine may take too much time. In addition, distribution helps in fault tolerance, since the task is not dependent on a single machine and is resistant to single point failure.

At the end of the step, the machine learning model is built and saved for deployment. The deployed model is updated as required, which is controlled by the adaptation service. We provide further details of our approach in the upcoming chapters.

CHAPTER 7

AUTHENTICATION

The authentication service receives features as input, which are output from the feature extraction algorithm to pass on as input to the deployed machine-learning model. A decision is output from the model next. The output from the model determines what the next piece of action would be for the authentication service. Depending on the output, if the user is determined to be a legitimate user, the authentication service continues to run in the background; otherwise, the authentication service locks the device and forces the user to re-authenticate herself.

On a regular basis, all the steps of our approach from data collection (step 1) to authentication (step 4) work in conjunction to drive the continuous authentication scheme. Whereas, periodically, depending on the feedback received by the adaptation service from the authentication service or otherwise as required, the adaptation step comes into play. Though at first glance it might so appear that the step of authentication is the last step of our approach, it is not in actual. In fact, the feedback from the authentication service to an extent determines when the model gets updated.

The performance of prediction of the deployed model is noted and shared with the adaptation service. If the performance of the model based on the evaluation metrics deteriorates by a substantial value, then the adaptation service updates the model. The updated model replaces the existing model and is made available to the authentication service as the newly deployed model.

It is worthy of mention however, that the focus of this thesis is not on when to begin the process of adaptation, rather on how to adapt and update the model. We provide a brief idea regarding how the feedback from the authentication service leads to adaptation up next, however, that may not be the sole driving factor to initiate the process of adaptation. In this thesis, we focus more on how efficiently adaptation can be carried out. A detailed research on when to adapt is left as a future work.

The feedback from the authentication service prompts the adaptation service to update the model. To understand the process of generating a feedback consider a scenario as follows: Let a user re-authenticate herself during the current usage session as she is determined to be an illegitimate user. If during the present usage session, the user frequently needs to re-authenticate herself, and if this happens more than a certain number of times, then according to our approach, the user needs to re-authenticate herself using a stronger means of authentication which can be a one-time password(OTP). If after this step, a user is determined to be a legitimate user, feedback in the form of information about the collected data and features is shared with the adaptation service. However, if the user is determined to be an illegitimate user, at this point, no further action is taken. Also, if the user is determined to be an illegitimate user once or twice during a usage session, feedback information is sent to the adaption service. The evaluation metrics for multiple instances of data is noted, which cumulatively provide us the value for evaluation metrics. For example, accuracy percentage is noted for multiple data instances and that value is considered a feedback parameter, which is shared with the adaptation service.

CHAPTER 8

ADAPTATION

User behavior may evolve over time. The way you handle your mobile device the day you start using it should be slightly different from the way you use it as a regular user. This is an example of user behavioral evolution. A straightforward solution is to capture user behavioral evolution is to retrain the model on the new data. There are two disadvantages of this approach. Firstly, the old model is completely wasted, since it is not re-utilized in the retraining process. Secondly, we miss important information of user characteristics that the previous dataset could have captured if we retrain the model just on new data. With "new data", we refer to data that has not been used in the past for training purposes. For example, the user may start following a new work out regime. She may revert to the old workout exercises few days later. Just depending on new data would not help the predictive performance in such a case. A solution is to retrain on all the available data collected so far. Even then, we cannot make use of the previously built model. We would simply waste that. In fact, building a new model on a larger data is an expensive choice in terms of the time complexity and the computational resources used.

We present a novel approach of continuous learning in this thesis. In this method, the boosted decision tree model keeps the previous model and continues training with the new incoming training data so that you can further boost an already fitted model on new data. With this approach, one would spend as many resources as needed to spend to train just the new data. For example, consider you have a dataset with 200 data points. You now build a model by training with 150 data points. Now, with our approach you can train the remaining dataset of 50 data points on top of the built model (with training data of size 150). This saves computation cost of building a brand-new model and can instead utilize the existing model and update the existing model. This is because you do not need to train a model again with 200 data points. Just training on the 50 data points on top of the already built model would efficiently serve the purpose. This is a method of continuous learning or continuous training. We plan to apply this method when the accuracy of the model drops substantially.

With gradient boosting, it is possible to further boost the already trained model while training it on new data. In the chapter of training the tree model, we discussed how the gradient boosted decision tree model gets trained by fitting a new tree to the residue. While updating the model, the new data becomes the residue, on which a new tree(s) is fitted and recursively the algorithmic steps from Algorithm 3 are repeated. The process of updating the model utilizes just as many computational resources as it is required to train on the new data. Clearly, this method is computationally more efficient than retraining the machine-learning model from scratch, since training models is extremely costly in terms of the acquired computational time and resources.

Overloading the gradient boosted decision tree model algorithm of our approach with too much data may increase computational cost, and may also lead to overfitting. It is important to update the model carefully to ensure that the built model does not overfit the data, which might in turn reduce the efficiency of the model. When to adapt, could be an area of research and we leave it more as a future work.

Since, too much data can do our approach more harm than good, we utilize a method of sampling to selectively use samples of data, rather than the entire dataset to reduce the data load on our model. This method is presented as an alternative to using the entire dataset for updating the model time and again. We utilize a method of stratified sampling as a means of selecting samples from the dataset. Stratified sampling randomly selects data from strata or groups from the dataset. It is more advantageous than random sampling because underrepresentation and overrepresentation of a group of data can be prevented, since equal number of samples get selected from each group of data. Furthermore, stratified sampling generally requires fewer samples than random sampling. We choose fixed number of samples from each window, which we consider as groups. The concept of "windows" was discussed in the chapter on feature extraction. We selected the windows as "groups" for stratified sampling as we found that to be the most basic element of our dataset which is omnipresent, especially since the data for our approach is more of a time-series data. The method of sampling may or may not be incorporated until a period as required depending on how many data samples are available in total. We provide further discussion on stratified sampling in the next chapter.

We plan to do continuous training periodically. But, after a period, we propose to build a new model and discard the existing model. This can be executed after a few months or so. This is still better than regular re-building of the model at more frequent intervals. Rebuilding a model after a period would help us capture and consider more recent data since at the point of regenerating the model, older data may be discarded. Data from wearables get stored only for a certain period, which can be considered a safe estimate to work with one model. After the end of the period or when we are halfway through the period, we can generate a new model.

CHAPTER 9

CASE STUDY OF OUR APPROACH

9.1 Data Collection

We use the Pervasive Systems Research Group's Sensor Activity Dataset [46], which we refer to as the PSRG dataset and the MHealth dataset [47, 48] to train our model. The PSRG data has been recorded at the rate of 50 samples per second. The MHealth dataset on the other hand has been sampled at a sampling rate of 50 Hz. Data was recorded for 10 users of diverse profiles for both the datasets. The datasets provide the accelerometer and gyroscope readings for each of the 10 users in separate files.

We executed three distinct sets of experiments. Each experiment implements our approach with a different data set. We use three sets of data from the MHealth and the PSRG dataset together, which are as follows: Mhealth data with wearable sensors attached to wrist; PSRG data with smart phone sensors attached to wrist and waist. These three sets of data are referred to as MHealth Wrist, PSRG Wrist and PSRG Waist data respectively in this thesis. For the three datasets, we use the following sensor data readings:

- Accelerometer reading in x-axis
- Accelerometer reading in y-axis
- Accelerometer reading in z-axis
- Gyroscope reading in x-axis
- Gyroscope reading in y-axis
- Gyroscope reading in z-axis

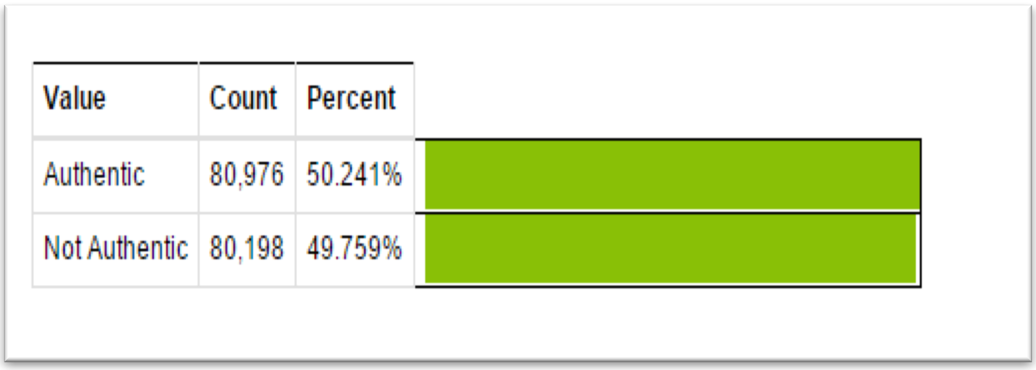
9.2 Feature Extraction

As described in the chapter of feature extraction, before extracting features, the datasets for the three sets of experiments undergo data preparation. Beyond which, features are extracted from the corresponding datasets for each user as per Algorithm 1, which is the feature extraction algorithm for our approach.

9.3 Data Preparation for Training the Tree Model

A training set with minimum bias [50] is considered best for learning and benchmarking. With bench marking, we mean performing better than random guessing. If there is equal probability of occurrence of two classes in a sample, then a machine learning model must perform better than random guessing performance, which is 50%, given there is equal chance of occurrence of each class label. A training set in which there is equal chance of occurrence of each label is called *unbiased*. To reduce bias thus and for better evaluation of our model, we build training datasets in a way as defined in the next paragraph.

To prepare unbiased training sets for training the tree model, we take approximately the same number of *illegitimate* user samples as the *legitimate* user samples. To do so, we consider one user to be legitimate and attach the class label "authentic" to her data samples. We consider the other 9 users illegitimate and attach the label "not-authentic" to their data samples. We take approximately equal number of samples of "not-authentic" label, such that it is approximately equal to the number of samples of the "authentic" label. Figure 9 shows the distribution of the class labels



Value	Count	Percent	
Authentic	80,976	50.241%	🟩
Not Authentic	80,198	49.759%	🟩

Fig 9: Training Set with Class Label Distribution

in a typical training set for our approach. This is consistent with all the three sets of experiments we executed.

As we have 10 distinct users for each of the three sets of experiments, we consider one user to be legitimate and the remaining 9 to be illegitimate and we repeat the process for all the 10 users. Thus, we get 10 distinct datasets with attached class labels for each of the three experiments. In other words, we prepare 10 sets of data out of each of the datasets (MHealth Wrist, PSRG Wrist and PSRG Waist data). Once we have all the datasets prepared in this way, we split the sets in a way such that 60-70% of the data is kept for training and the remaining is kept for testing. 5% of the data from the training set is reserved for validation set testing.

The purpose of having 50% of each of the class labels is to reduce bias in the training set. However, having 10 different sets of data for forming the train and test datasets is not required to reduce bias. In fact, running our experiments on different sets of training and test data helps us test our approach well. In this way, we can test user behavioral profile for 10 different users, in which case, each user is considered legitimate once. So, in a nutshell, with this approach, we are able to study the behavioral profile of 10 users where activities of one user is considered to be legitimate and the activities of the other user is considered illegitimate.

9.4 Feature Selection and Training the Tree Model

The model learns from the training set. As discussed earlier, the feature selection and the model generation steps are combined in our approach for better optimization. This also ensures that we do not lose any important information in the process. This also prevents additional overhead of having a separate feature selection step. Decision tree algorithm splits a node based on its feature importance and thus it optimizes and combines the feature selection and the model training steps into one step saving computational time and resources. We have observed that the top features vary with users. In an attempt to make our approach more personalized, instead of selecting the same set of features for all the users, it's a good idea to do feature selection as part of model generation, where the decision tree algorithm selects the top features for each user.

The tree model is trained on the training sets as described by the algorithms described in the chapter on training the gradient boosted tree model. The training and the test sets are prepared as described in the earlier sections. At the end of this step, the tree model is built and it is evaluated on the test sets. This process is repeated for the three sets of experiments.

For our experiments, the PSRG based training sets comprised of 85,000 data samples on an average and the test sets had approximately 38,000 samples. The number of samples for each train-test pair varied for the MHealth group based datasets. The number of samples for the train sets based on MHealth data ranged from

111,000 to 160,000 samples. The number of samples for the test sets ranged from 78,000 to 103,900 samples.

9.5 Tree-Model Evaluation

To evaluate the predictive performance for each model, we consider evaluation metrics like accuracy, EER (Equal Error Rate), AUC (Area under curve of the ROC curve), precision, recall, f1 score, log loss amount etc. The evaluation results are provided in the next section for each of the three sets of experiments. As discussed earlier, for each experiment, 10 pairs of train-test datasets were prepared. For each pair, the machine-learning model was trained using the training set and the built model was evaluated on the test set. The evaluation result values for each of the evaluation metrics across all the test sets were averaged to provide the listed evaluation metric values in Table 4. Each built machine-learning model was evaluated by considering an average of the cross-validation based evaluation results. The values for each of the evaluation metrics were comparable across all the train-test pairs per experiment, and the cross validation based results were also consistently comparable to each other.

9.6 Results

We present the results as obtained as part of running the experiments in this section. Table 4 provides the evaluation results. Refer to Figure 10 for the ROC curve

Table 4: Evaluation Results

Evaluation Criteria	PSRG Wrist	PSRG Waist	MHealth Wrist
Accuracy	99%	98.6%	97%
AUC	1.0	0.9972	0.996
EER	<1%	1%	1%
f-1 score	0.98	0.97	0.975
Log Loss	0.10	0.13	0.1575
Precision	0.99	0.99	0.98
Recall	0.97	0.97	0.95

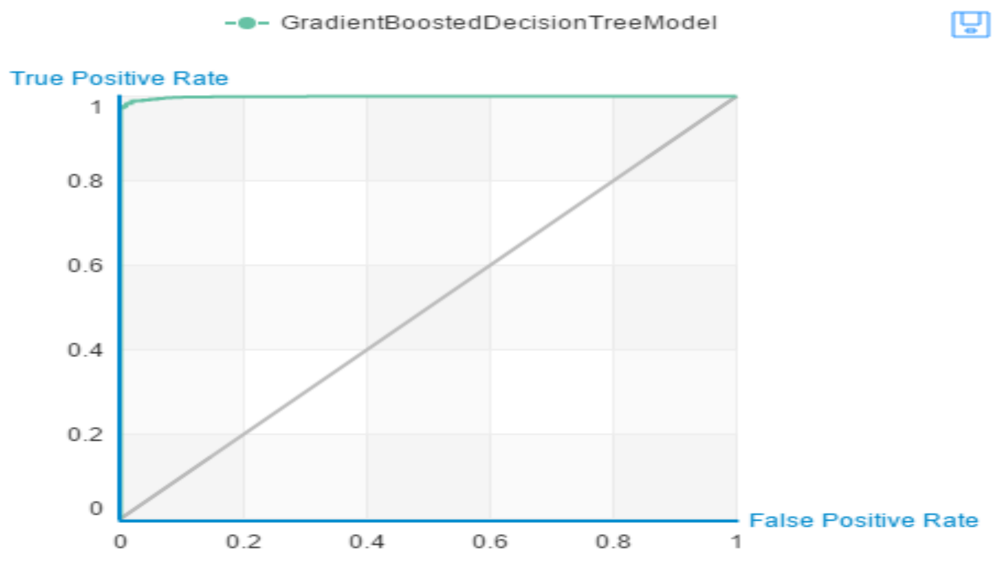


Fig 10: ROC curve for the Tree Model of our approach

for the generated machine-learning model. Since, the results per user are comparable without significant variation, we show the ROC curve for one machine-learning model and consider it to be representative of all the other generated machine-learning models.

9.7 Comparison

The performance of our authentication method, more specifically the machine-learning model generation algorithm is compared to the machine-learning model generation techniques presented in few recent papers in this section. These techniques are designated as follows: Logistic Regression (LR) [49], Random Forests (RF) [33], Support Vector Machines (SVM) [25, 34, 60], and Decision Trees (DT) [60]. It is important to note that according to the paper [60], which used decision trees and SVM, and achieved the accuracy of up to 98%, the number of test samples used was just 100. Whereas, in our approach, we used approximately 38,000 test samples for the PSRG based experiments and at least 90,000 test samples for the MHealth based experiments. We chose to not implement KNN due to the high query complexity of the KNN algorithm [13]. We observed that SVM achieves a very accuracy, but, at the cost of quadratic time complexity. Table 5 provides a comparison of how models built from different techniques perform as compared to our approach.

Refer to Figure 11 for the comparison of the ROC curves for the three methods of tree-based classifiers: our approach, decision trees [60], random forests [13]. The ROC (Receiver Operating Characteristics) curve in statistics is a graphical plot that explains the performance of a binary classifier system as its prediction threshold is

Table 5: Comparison with Existing Methods

Algorithms	Reported Accuracy	Mean Accuracy (Our approach)
LR [6]	82.30%	82.88%
RF [13]	-	85.5%
SVM [25]	Average – 90%, Maximum – 95%	94%
SVM [60]	* 98%	94%
DT [60]	* 98%	92%

** number of test samples = 100.*

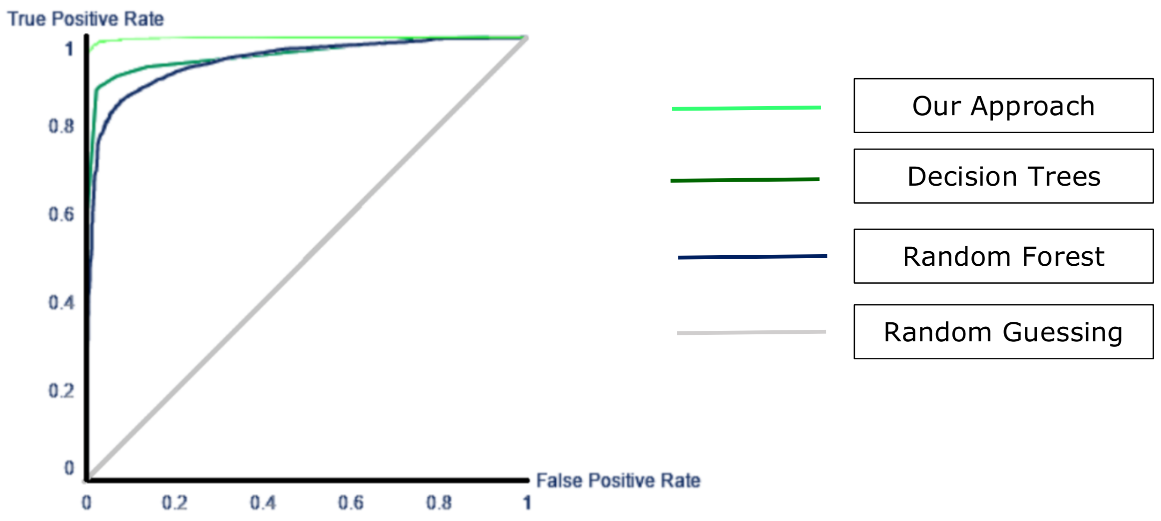


Fig 11: ROC Curve Comparison

modified. The ROC curve provides nuanced insights about the conduct of the classifier. The curve is drawn by plotting the true positive rate against the false positive rate at various threshold settings. In other words, it illustrates how many correct positive classifications can be obtained as more and more false positives appear. The area under the curve (AUC) computes the probability that the binary classifier will rank a randomly chosen positive sample higher than a randomly chosen negative sample. The perfect classifier would have AUC as 1. So, the more the AUC, the better the classifier at making reliable predictions. In our case, the *"authentic"* user label is considered the positive label. From Figure 11, we see that with our approach, the AUC is maximum and much more than other similar methods such as decision trees and random forests. The diagonal line in Figure 11 represents the ROC curve for random guessing. A machine-learning algorithm must perform better than at least random guessing. Clearly, all the three curves are above the diagonal line and have greater AUC. The AUC for a line, which passes through the diagonal, is 0.5. A classifier should at least have the AUC value of 0.5 thus. It is considered that an AUC of 1 represents the perfect classifier, and an AUC of 0.5 represents a worthless classifier. Thus, the AUC value varies from 0.5 to 1. The AUC value of our approach is 0.99 approximately, which is better than the existing methods of model generation.

9.8 Adaptation

In order to carry out the experimentation on adaptation, we select 60% of the data available. We call this the initial data. We use 70% of the initial data for training and 30% for testing while building the model. We call the built model the initial model. Next, we continue training on the remaining data to further boost the trained model.

Table 6: Stratified Sampling as a means of Data Sampling

Criteria	Without Sampling	Random Sampling	Our Approach – Stratified Sampling
Definition	Does not select samples from data	Randomly select samples from data	Select samples from each group
Data Dependency	Uses 100% of data	Uses 60% of data	Uses 42% of data
Training time	Takes more time	Takes less time	Takes least time

We utilize Gradient Boosted Decision Trees (GBDT) model to implement the continuous training approach. GBDT allows continuous training with new data to further boost an already fitted model.

We observe that the updated model has comparable performance to that of a model built using the entire data at one go. Additionally, we see that the continuous training approach takes approximately as much time as it would need to train just the new data. This is per the characteristics of gradient boosting whose time complexity scales in linear time with increase in the size of data.

Table 6 shows the how stratified sampling performs as a means of data sampling. Stratified sampling was discussed in the chapter on adaptation. We see that with around 40% of the data, it provides approximately the same accuracy on test data. However, it is worth pointing out that since the size of the datasets is not very large, we could not perform extensive analysis on the utility of stratified sampling, which is left as a future work.

9.9 Running Our Approach as a Service

We put to test our approach by writing the code for a service which is essentially the authentication service, running on an Android smart phone. The designed service received data as input and invoked the machine-learning model to output a decision. The decision was displayed as a result.

9.10 Discussion

Traditional password based methods of authentication usually require the user to remember passwords or come up with a complex password for better protection. In this thesis, we present an approach to generate a user-centric machine-learning model by capturing motion gestures of the user. In our approach, the user is not required to follow a specific script to exhibit gestures. We validate the user again on motion gestures without needing user participation. Users do not need to exhibit complex gestures as well for the same reason. Users can exhibit gestures as they wish. We obtain high predictive performance by following the presented approach.

Physiological biometric based authentication systems require additional state of the art sensors to authenticate the user. We utilize sensors that are pre-embedded in the wearable device, thus eliminating the need of additional sensors. Additionally, motion gestures are recorded for model generation and user validation and throughout the continuous authentication process conscious user inputs are not required, making the authentication process unobtrusive to a legitimate user. Only at the time when a user is determined to be an illegitimate user, the user is forced to re-authenticate herself.

We get complete motion information of the body part the wearable is attached to by using just the accelerometer and gyroscope sensors. It helps us better model the user gestures by using a combination of these two motion sensors as we can record complete motion data. Existing methods have also used magnetometer sensors. Magnetometer sensors firstly do not provide noticeable changes in the sensor readings if the authentic user and the attacker are in the same geographical location approximately. Secondly, many wearable devices do not have a magnetometer sensor embedded in them. Furthermore, we see that without using additional sensors we achieve high performance. Thus, we do not use the magnetometer sensor in our approach.

Gait pattern based user identification can authenticate the users only when the users are walking. As discussed earlier, our method does not impose any constraint on part of the user regarding the kind of activities she must perform to be correctly authenticated. A user centric approach is presented which adapts to user choices and preferences and updates the model to capture user behavioral evolution.

We do not need an additional feature selection step in our approach since gradient boosting on decision trees automatically does that for us. This makes our model more versatile and robust since we do not lose any information at any point before the model generation step. Since, our approach is user centric, this method of model generation helps the model learn the user feature better and be specific to a user. Thus, we build a far-reaching, user centric approach, which adapts to user preferences and behavior.

Our approach also provides higher performance by implementing a model with lesser time complexity. Some of the existing works have achieved high performance using SVM and KNN, which are inherently more complex. SVM has quadratic time complexity, whereas KNN depends on a complex query step. In our approach, the machine learning model is generated by implementing gradient boosting on decision trees, which scales in linearithmetic time – O (nlogn) and has lower time complexity than most other existing algorithms, where *n* stands for the number of samples, which are used to train the model. Experimental results show that higher accuracy and lower EER than existing methods can be obtained with our approach.

Our approach also ensures that overfitting is minimized. Overfitting occurs when the generated machine-learning model fails to generalize beyond the training set. The process to update the model only requires as much resources and time as it is required to just train on newer data. With "new data", we refer to the data, which has not been utilized for training the previously generated model. The training is done on top of the existing model thus requiring less time and resources than it is required to build a new model from scratch. Furthermore, we see that our approach can be extended to the smart phone use case as well, provided the smart device is attached to the body part of the user or is in contact with the user.

CHAPTER 10

CONCLUSION AND FUTURE WORK

In this thesis, we presented an innovative approach to develop an authentication software by modeling user movement gestures for continuous authentication of smart wearable users. Sensor data from the wearable device is utilized in this approach, eliminating the need for additional hardware. This approach incorporates continuous learning to adapt to user motion gesture evolution. Our approach does not need conscious inputs from the user to execute, and we do not impose a restriction on the kind of motion gestures the user must exhibit to be correctly authenticated. We see that our approach produces better results with lower time complexity than existing methods. We observe that using complete motion gesture information with data from accelerometers and gyroscopes, better performance is achieved.

As a future work, we plan to implement an application, which implements our approach end-to-end. For this we plan to train the model on remote cloud servers and deploy the model on the smart wearable device. We believe this is possible, since the model only takes as much space as an ordinary application running on the device would take. An authentication service would invoke a feature extraction unit with live sensor data from the user. The feature extraction unit after computing feature values would invoke the machine-learning model and pass in the features as input. The response from the model would be sent to the authentication service. The authentication service depending on the response would lock the device forcing the user to re-authenticate herself provided the user is determined to be an illegitimate user. Otherwise, it would continue to run in the background. We also plan to test our model on additional sets of users with diverse profiles.

Furthermore, as discussed earlier, we would like to research further on when an update to the deployed machine-learning model is suitable. Also, we would like to conduct extensive experiments on the suitability of stratified sampling for our approach.

REFERENCES

[1] Thierer, A. D. (2015). The internet of things and wearable technology: Addressing privacy and security concerns without derailing innovation.

[2] IDC. (2015, December 17). *IDC Forecasts Worldwide Shipments of Wearables to Surpass 200 Million in 2019, Driven by Strong Smartwatch Growth - prUS40846515*. Retrieved March 28, 2017, from http://idc.com/getdoc.jsp?containerId=prUS40846515

[3] Apple Watch. (2017, March 17). In *Wikipedia, The Free Encyclopedia*. Retrieved 01:03, March 29, 2017, from https://en.wikipedia.org/w/index.php?title=Apple_Watch&oldid=770843458

[4] Lumo. (2016). *About Lumo Bodytech | Lumo*. Retrieved March 29, 2017, from http://www.lumobodytech.com/about/

[5] Fitbit. (2017, March 7). In *Wikipedia, The Free Encyclopedia*. Retrieved 01:05, March 29, 2017, from https://en.wikipedia.org/w/index.php?title=Fitbit&oldid=769030997

[6] Myo armband. (2017, January 21). In *Wikipedia, The Free Encyclopedia*. Retrieved 04:07, March 29, 2017, from https://en.wikipedia.org/w/index.php?title=Myo_armband&oldid=761113974

[7] Chiauzzi, E., Rodarte, C., & DasMahapatra, P. (2015). Patient-centered activity monitoring in the self-management of chronic health conditions. *BMC medicine*, *13*(1), 77.

[8] Khan, R., Hasan, R., & Xu, J. (2015, March). SEPIA: secure-PIN-authentication-as-a-service for ATM using mobile and wearable devices. In *Mobile Cloud Computing, Services, and Engineering (MobileCloud), 2015 3rd IEEE International Conference on* (pp. 41-50). IEEE.

[9] Multi-factor authentication. (2017, March 25). In *Wikipedia, The Free Encyclopedia*. Retrieved 20:31, March 29, 2017, from https://en.wikipedia.org/w/index.php?title=Multi-factor_authentication&oldid=772169051

[10] Weisbaum, H. (2014, April 26). *Most Americans don't secure their smartphones*. Retrieved March 29, 2017, from

http://www.cnbc.com/2014/04/26/most-americans-dont-secure-their-smartphones.html

[11] Ben-Asher, N., Kirschnick, N., Sieger, H., Meyer, J., Ben-Oved, A., & Möller, S. (2011, August). On the need for different security methods on mobile phones. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services* (pp. 465-473). ACM.

[12] Zakaria, N. H., Griffiths, D., Brostoff, S., & Yan, J. (2011, July). Shoulder surfing defence for recall-based graphical passwords. In *Proceedings of the Seventh Symposium on Usable Privacy and Security* (p. 6). ACM.

[13] Aviv, A. J., Gibson, K. L., Mossop, E., Blaze, M., & Smith, J. M. (2010). Smudge Attacks on Smartphone Touch Screens. *Woot*, *10*, 1-7.

[14] Devine, R. (2012). *Face Unlock in Jelly Bean gets a 'Liveness check' | Android Central*. Retrieved March 29, 2017, from http://www.androidcentral.com/face-unlock-jelly-bean-gets-liveness-check

[15] Shaji, S., Das, S., & Kizhakkethottam, J. J. (2015, February). Review of continuous touch based user authentication. In *Soft-Computing and Networks Security (ICSNS), 2015 International Conference on* (pp. 1-5). IEEE.

[16] *CCC | Fingerprint Biometrics hacked again*. (n.d.). Retrieved March 29, 2017, from http://www.ccc.de/en/updates/2014/ursel

[17] Dorflinger, Tim, et al. "“My smartphone is a safe!” The user's point of view regarding novel authentication methods and gradual security levels on smartphones." *2010 International Conference on Security and Cryptography (SECRYPT)*. 2010.

[18] Chang, T. Y., Tsai, C. J., & Lin, J. H. (2012). A graphical-based password keystroke dynamic authentication system for touch screen handheld mobile devices. *Journal of Systems and Software*, *85*(5), 1157-1165.

[19] Nixon, K. W., Chen, Y., Mao, Z. H., & Li, K. (2014). User classification and authentication for mobile device based on gesture recognition. In *Network Science and Cybersecurity* (pp. 125-135). Springer New York.

[20] Hestbek, M. R., Nickel, C., & Busch, C. (2012, April). Biometric gait recognition for mobile devices using wavelet transform and support vector machines. In *Systems, Signals and Image Processing (IWSSIP), 2012 19th International Conference on* (pp. 205-210). IEEE.

[21] Buduru, A. B., & Yau, S. S. (2015, August). An effective approach to continuous user authentication for touch screen smart devices. In *Software Quality,*

Reliability and Security (QRS), 2015 IEEE International Conference on (pp. 219-226). IEEE.

[22] Amitay, D. (2011, June 14). *Most Common iPhone Passcodes — Daniel Amitay*. Retrieved March 31, 2017, from http://danielamitay.com/blog/2011/6/13/most-common-iphone-passcodes

[23] Starner, T., & Martin, T. (2015). Wearable computing: The new dress code [guest editors' introduction]. *Computer*, *48*(6), 12-15.

[24] Yang, J., Li, Y., & Xie, M. (2015, March). Motionauth: Motion-based authentication for wrist worn smart devices. In *Pervasive Computing and Communication Workshops (PerCom Workshops), 2015 IEEE International Conference on* (pp. 550-555). IEEE.

[25] Lee, W. H., & Lee, R. B. (2015, February). Multi-sensor authentication to improve smartphone security. In *Information Systems Security and Privacy (ICISSP), 2015 International Conference on* (pp. 1-11). IEEE.

[26] Marforio, C., Karapanos, N., Soriente, C., Kostiainen, K., & Capkun, S. (2014, February). Smartphones as Practical and Secure Location Verification Tokens for Payments. In *NDSS*.

[27] Cornelius, C., & Gutierrez, C. N. (2014). A SURVEY OF BIOMETRICS FOR WEARABLE DEVICES. *Intel Technology Journal*, *18*(4).

[28] Zhu, J., Wu, P., Wang, X., & Zhang, J. (2013, January). Sensec: Mobile security through passive sensing. In *Computing, Networking and Communications (ICNC), 2013 International Conference on* (pp. 1128-1133). IEEE.

[29] Qi, M., Lu, Y., Li, J., Li, X., & Kong, J. (2008, December). User-specific iris authentication based on feature selection. In *Computer Science and Software Engineering, 2008 International Conference on* (Vol. 1, pp. 1040-1043). IEEE.

[30] Hong, L., & Jain, A. (1998). Integrating faces and fingerprints for personal identification. *IEEE transactions on pattern analysis and machine intelligence*, *20*(12), 1295-1307.

[31] Negara, A. F. P., Kodirov, E., Abdullah, M. F. A., Choi, D. J., Lee, G. S., & Sayeed, S. (2012). Arm's flex when responding call for implicit user authentication in smartphone. *Int. J. Secur. Its Appl*, *6*(879), 83.

[32] Lu, H., Huang, J., Saha, T., & Nachman, L. (2014, September). Unobtrusive gait verification for mobile phones. In *Proceedings of the 2014 ACM international symposium on wearable computers* (pp. 91-98). ACM.

[33] Davidson, S., Smith, D., Yang, C., & Cheah, S. (2016). Smartwatch User Identification as a Means of Authentication. *Department of Computer Science and Engineering Std*.

[34] Chauhan, J., Asghar, H. J., Kaafar, M. A., & Mahanti, A. (2014). Gesture-based continuous authentication for wearable devices: the google glass case. *arXiv preprint arXiv:1412.2855*.

[35] Yang, K. (2011). Eliza Yingzi Du,"Consent Biometrics".

[36] Gartner. (2014). *Gartner Says Worldwide Smartwatch and Wristband Market Is Poised for Take Off*. Retrieved April 1, 2017, from http://www.gartner.com/newsroom/id/2848817

[37] Mariani, B., Jiménez, M. C., Vingerhoets, F. J., & Aminian, K. (2013). On-shoe wearable sensors for gait and turning assessment of patients with Parkinson's disease. *IEEE transactions on biomedical engineering*, *60*(1), 155-158.

[38] Cisco. (2011). How Secure are Numeric Passwords? Retrieved April 2, 2017, from http://blogs.cisco.com/security/how-secure-are-numeric-passwords

[39] Sra, S. (2011). Fast projections onto ℓ 1, q-norm balls for grouped feature selection. *Machine learning and knowledge discovery in databases*, 305-317.

[40] Duchi, J., Shalev-Shwartz, S., Singer, Y., & Chandra, T. (2008, July). Efficient projections onto the l 1-ball for learning in high dimensions. In *Proceedings of the 25th international conference on Machine learning* (pp. 272-279). ACM.

[41] Hastie, T., Tibshirani, R., & Friedman, J. (2009). Springer series in statistics. *The elements of statistical learning: data mining, inference, and prediction*.

[42] Sugumaran, V., Muralidharan, V., & Ramachandran, K. I. (2007). Feature selection using decision tree and classification through proximal support vector machine for fault diagnostics of roller bearing. *Mechanical systems and signal processing*, *21*(2), 930-942.

[43] Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, 1189-1232.

[44] Freund, Y., Schapire, R., & Abe, N. (1999). A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, *14*(771-780), 1612.

[45] Johnson, R., & Zhang, T. (2014). Learning nonlinear functions using regularized greedy forest. *IEEE transactions on pattern analysis and machine intelligence*, *36*(5), 942-954.

[46] Shoaib, M., Bosch, S., Incel, O. D., Scholten, H., & Havinga, P. J. (2014). Fusion of smartphone motion sensors for physical activity recognition. *Sensors*, *14*(6), 10146-10176.

[47] Banos, O., Garcia, R., Holgado-Terriza, J. A., Damas, M., Pomares, H., Rojas, I., ... & Villalonga, C. (2014, December). mHealthDroid: a novel framework for agile development of mobile health applications. In *International Workshop on Ambient Assisted Living* (pp. 91-98). Springer International Publishing.

[48] Banos, O., Villalonga, C., Garcia, R., Saez, A., Damas, M., Holgado-Terriza, J. A., ... & Rojas, I. (2015). Design, implementation and validation of a novel open framework for agile development of mobile health applications. *Biomedical engineering online*, *14*(2), S6.

[49] Primo, A., Phoha, V. V., Kumar, R., & Serwadda, A. (2014). Context-aware active authentication using smartphone accelerometer measurements. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (pp. 98-105).

[50] Dietterich, T. G., & Kong, E. B. (1995). *Machine learning bias, statistical bias, and statistical variance of decision tree algorithms*. Technical report, Department of Computer Science, Oregon State University.

[51] Ridgeway, G. (2007). Generalized Boosted Models: A guide to the gbm package. *Update*, *1*(1), 2007.

[52] Bianchi, A., & Oakley, I. (2016). Wearable authentication: Trends and opportunities. It-Information Technology, 58(5), 255-262. 77

[53] Morphy, E. (2013, June 03). Google Glass Drops Facial Recognition (For Now). Retrieved May 12, 2017, from https://www.forbes.com/sites/erikamorphy/2013/06/02/google-glass-drops-facial-recognition-for-now/#17b883fb4e38

[54] C4.5 algorithm. (2016, June 27). In Wikipedia, The Free Encyclopedia. Retrieved 02:41, May 19, 2017, from https://en.wikipedia.org/w/index.php?title=C4.5_algorithm&oldid=727181226

[55] Nymi. (2017). Product Overview | Nymi. Retrieved June 7, 2017, from https://nymi.com/product_overview

[56] Nanavati, S., Thieme, M., & Nanavati, R. (2002). *Biometrics: identity verification in a networked world*. New York: Wiley.

[57] De Luca, A., Hang, A., Brudy, F., Lindner, C., & Hussmann, H. (2012, May). Touch me once and i know it's you!: implicit authentication based on touch screen patterns. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 987-996). ACM.

[58] Bonastre, J. F., Bimbot, F., Boë, L. J., Campbell, J. P., Reynolds, D. A., & Magrin-Chagnolleau, I. (2003). Person authentication by voice: A need for caution. In *Eighth European Conference on Speech Communication and Technology*.

[59] Low, Y., Bickson, D., Gonzalez, J., Guestrin, C., Kyrola, A., & Hellerstein, J. M. (2012). Distributed GraphLab: a framework for machine learning and data mining in the cloud. *Proceedings of the VLDB Endowment*, *5*(8), 716-727.

[60] Sugimori, D., Iwamoto, T., & Matsumoto, M. (2011, August). A study about identification of pedestrian by using 3-axis accelerometer. In *Embedded and Real-Time Computing Systems and Applications (RTCSA), 2011 IEEE 17th International Conference on* (Vol. 2, pp. 134-137). IEEE.