

April 2017

A Behavioral Model System for Implicit Mobile Authentication

Arthur Joseph Dooner
Worcester Polytechnic Institute

Arun Nagaraja Donti
Worcester Polytechnic Institute

Stephen Robert Lafortune
Worcester Polytechnic Institute

Walter T. Ho
Worcester Polytechnic Institute

Follow this and additional works at: <https://digitalcommons.wpi.edu/mqp-all>

Repository Citation

Dooner, A. J., Donti, A. N., Lafortune, S. R., & Ho, W. T. (2017). *A Behavioral Model System for Implicit Mobile Authentication*. Retrieved from <https://digitalcommons.wpi.edu/mqp-all/1792>


This Unrestricted is brought to you for free and open access by the Major Qualifying Projects at Digital WPI. It has been accepted for inclusion in Major Qualifying Projects (All Years) by an authorized administrator of Digital WPI. For more information, please contact digitalwpi@wpi.edu.

A Behavioral Model System for Implicit Mobile Authentication

A Major Qualifying Project
Submitted to the Faculty of
WORCESTER POLYTECHNIC INSTITUTE
In partial fulfillment of the requirements for the
Degree in Bachelor Science
In
Computer Science
By


Arun Donti


Arthur Dooner


Walter Ho


Stephen Lafortune

This report represents the work of WPI undergraduate students submitted to the faculty as evidence of completion of a degree requirement. WPI routinely publishes these reports on its website without editorial or peer review. For more information about the project program at WPI, please see <http://www.wpi.edu/academics/ugradstudies/project-learning.html>

Date: April, 2017
Project Advisor:

Prof. Emmanuel Agu, Advisor

Abstract

Smartphones are increasingly pervasive in users' everyday lives. Security concerns of data compromises are growing, and explicit authentication methods such as passwords are proving to be inconvenient and inadequate. To address this, behavioral authentication approaches have been proposed wherein a user is authenticated continuously and implicitly, by utilizing consistent patterns in their behavior. This research project develops a Behavioral Model System (BMS) that records users' behavioral metrics on an Android device and sends them to a server to develop a behavioral model for the user. Once a strong model is generated using deep learning, a user's most recent behavior is queried against the model to authenticate them, 2 out of 4 test subjects had unique behaviors that identified them.

Contents

Abstract.....	1
1. Introduction.....	6
1.1. Authentication Methods.....	6
1.2. Importance of the Modern Smartphone	7
1.3. Current Security Measures on Smartphones.....	8
1.4. Limitations of Current Security Methods.....	11
1.5 Behavioral Authentication.....	12
1.6 The Goal of This Project.....	13
2. Background	15
2.1 Data Smartphones Collect	15
2.2 Doze: Android's Power Saving Mechanism	17
2.3. Deep Learning/Neural Networks	20
3. Related Work.....	23
3.1 Comparative Research on Positive Identification	23
3.2 Generalized Algorithm for Categorizing Behavioral Authentication.....	27
3.3 Deep Neural Nets for Modeling Mobile Soft Keystroke Authentication.....	28
3.4 Touchalytics	29
3.5 Soft Authentication with Low-Cost Signatures	31
3.6 Continuous Authentication on Mobile Devices Using Power Consumption, Touch Gestures and Physical Movement of Users.....	32
3.7 Authentication Feature and Model Selection using Penalty Algorithms	33
3.8 Miscellaneous	34
3.9 Summary	34
4. Methodology.....	36
4.1 Pilot Study.....	36
4.2 Assumptions	36
4.3 Metrics Selected for Inclusion in BMS	37
4.4 Gathering the Data.....	40
4.5 Privacy Concerns.....	41
4.6 Authentication	41
5. BMS System Implementation	43
5.1 Implementing Our Authentication Modality	44

5.2 Implementing Data Collection as a Background Process	45
6 Results	51
6.1 Discussion of Behavior of Users Surveyed.....	51
6.2 Battery Tests.....	52
6.3 Authentication Results	52
6.4 Individual Metrics	59
6.5 Discussion of Results.....	67
7 Future Works.....	69
7.1 Modular Implementation Allowing User to Select Various Combinations of Metrics.....	69
7.2 Negative Sample Models	69
7.3 Multi-User Support on Shared Devices	70
7.4 Distinguishing Group of Friends with Many Shared Behavioral Patterns.	70
7.5 Multi-Device Connection and Internet of Things	71
7.6 Evaluating Different Neural Networks.....	72
8 Conclusions.....	73
Citations.....	Error! Bookmark not defined.

Table of Tables

Table 1. Methods used to Authenticate Smartphones	9
Table 2. Vulnerabilities of Current Smartphone Authentication Methods	12
Table 3. Common Smartphone Hard Sensors[25].....	16
Table 4. Soft Sensor Features used by Rachuri et al [26].	17
Table 5. Some Common Neural Network Types [34].....	21
Table 6. A Comparison of Sensors, Methods, and Accuracy for Identification/Authentication[24].....	25
Table 7. Recognition, verification, and error rates of behavioral metrics	26
Table 8. The sensors we are tracking and the API's used to do it.	45
Table 9. Summary of data gathered from our selected behavioral metrics.	46
Table 10. Metrics implemented by BMS and Features extracted from them	49
Table 11. Battery drain with and without the BMS app.	52
Table 12. Day 11 Confusion Matrix of training on all metrics.	56
Table 13. List of most commonly used applications by Subject	59

Table of Figures

Figure 1. Wakelocks in Android with Doze (off, then turned on) [31].	19
Figure 2. An example neural network [33]	20
Figure 3. Smartphone strokes recorded for eight different users, showing unique user touch behavior	30
Figure 4. Features' effectiveness to identify an individual.	31
Figure 5. Feeding into the ANN for an authentication output	41
Figure 6. BMS design.	44
Figure 7. Wi-Fi, Tilt, App Usage, and Location on subject A's model.	54
Figure 8. Wi-Fi, Tilt, App Usage, and Location on subject B's model.	54
Figure 9. Wi-Fi, Tilt, App Usage, and Location on subject C's model	55
Figure 10. Wi-Fi, Tilt, App Usage, and Location on subject D's model.	55
Figure 11. App Usage on subject A's model.	60
Figure 12. App Usage on subject B's model.	60
Figure 13. App Usage on subject C's model.	61
Figure 14. App Usage on subject D's model.	61
Figure 15. Location on subject A's model.	62
Figure 16. Location on subject B's model.	62
Figure 17. Location on subject C's model.	63
Figure 18. Location on subject D's model.	63
Figure 19. Wi-Fi on subject A's model.	64
Figure 20. Wi-Fi on subject B's model.	64
Figure 21. Wi-Fi on subject C's model.	65
Figure 22. Wi-Fi on subject D's model.	65

1. Introduction

1.1. Authentication Methods

Authentication is the process by which a system verifies the identity of a user to allow access to it [1]. Authorization refers to the permissions and rules that determine the access that a user has [1]. On a college campus or at a workplace, an identification badge may authenticate a person, but it does not necessarily authorize him or her to access every building. These two concepts pertain to *allowing* access and the *level of* access granted to aspects of a secured system; however, both are central to the design of a security system. In a simple, single user system, these concepts have a *one-to-one relationship*. If a user's credentials are accepted by the system they are said to be "authenticated" and "authorized", otherwise they are both "unauthenticated" and "unauthorized". For the sake of this research, we assumed that authentication and authorization have this one-to-one relationship such that an authenticated user is also authorized to access the system. The process of authentication can be placed into two of four general subcategories based on both temporal attributes and authentication schemes. In time-based categories, authentication can either be Continuous or Episodic, while in scheme, authentication can either be Explicit or Implicit.

Definitions:

1. ***Continuous authentication:*** The process in which a system will constantly attempt to verify the identity of a user using a specific scheme [2].
2. ***Episodic authentication:*** The process in which a system will attempt to verify the identity of a user when the user's authorization status expires or is needed to perform a specific action [3].
3. ***Explicit authentication:*** The process in which a system will request users to specifically present credentials, to verify the identity of a user [4].

4. ***Implicit authentication:*** The process in which a system will verify a user through their behavior or actions [5].

Episodic authentication proves to be less computationally demanding in comparison to continuous authentication. However, Continuous authentication is inherently more secure as it authenticates in the same way as Episodic authentication, but runs more frequently. Explicit authentication is simpler to implement than Implicit authentication, as implicit requires significant research and overhead to program a device to understand how to distinguish between and authenticate the correct user. In theory, Implicit authentication is more secure than Explicit authentication as it offers more opportunities for authentication and is more difficult to spoof or replicate a person's behaviors and actions, rather than some sort of passcode [6].

Many systems used today follow the Episodic and Explicit authentication paradigm, usually implementing passwords, pins, and fingerprints. For instance, a website with personal information will generally ask for a username and password at the beginning, a smartphone might ask for a fingerprint, a campus might use a badging system to let personnel into a building and into specific rooms, and an important document might require a signature. Despite the intrigue and research opportunities of Implicit authentication, those approaches usually are regarded as too difficult to be a means of authentication in professionally-designed systems [7].

1.2. Importance of the Modern Smartphone

Over the past decade, smartphones have become ingrained into daily life, used for purposes ranging across gaming, social media, shopping, banking, email, and more. Estimates show that over two billion people were using smartphones worldwide by the end of 2016 [8].

Smartphones have sensors for measuring the user's environment and implicitly accumulate large volumes of personal data from frequent use - messages, pictures, documents, call logs, location traces, and usage logs. These data are unique to the user and can be viewed as their usage signature. Over the past few years, smartphone services have started to utilize this available data to improve functionality for the end user, for example, by remembering where you've parked your car.

Data from smartphones is growing easier to access simultaneously with their users interacting with them more. According to a study conducted by Time magazine in 2015, people in the United States checked their phones 46 times a day. That number rose to 74 times a day among users from ages 18 – 24 [9]. The 2015 Internet Trends report estimates that the average adult spent 5.6 hours a day on the internet with 2.8 of those hours being on a mobile device [10]. As the user interacts more with and relies more on their smartphone the data it accumulates is increasingly personal and sensitive. The phone collects email data, internet search data, access to social media accounts, bank accounts, and so much more.

1.3. Current Security Measures on Smartphones

Smartphones contain sensitive personal and non-personal data. Developers and researchers understand the personal nature of this locally-stored data, and have approached user authentication in a variety of ways. Table 1 lists some of the methods currently used to authenticate smartphones.

Method	Description
Pin	4 digit number the user enters to unlock the

	phone [11].
Password	A combination of letters, numbers, and symbols the user types in to unlock the phone [11].
Pattern	The user swipes a specific pattern to unlock their phone [11].
Pictures	The user touches/swipes a picture in the correct place(s) to unlock the phone [12].
Fingerprint	The user places one of their remembered fingers on the fingerprint scanner to unlock their phone [13, p. 0].
Iris	The phone will scan the user's eye to unlock the phone [14].
Trusted Devices	The phone will check the proximity of a trusted device (ex. smartwatch) through the Bluetooth connection to unlock the phone [15].
Voice Recognition	The phone recognizes the user's voice to unlock the phone [15].
NFC Tags	The user taps a paired object with an NFC tag on the phone to unlock it [15].
Location Based	The phone will automatically unlock if in a trusted location [15].

Table 1. Methods used to Authenticate Smartphones

Many of the authentication methods in Table 1 are episodic, requesting for credentials after certain timeframes, and explicit, requiring the user's direct inputs (typing in a password, holding their finger up to a fingerprint scanner). Several are used in conjunction with one another; for example, many phones with fingerprint scanners use a password or a pin as a backup

method if normal authentication fails, or in certain conditions specified by the user. Still in other situations, some authentication methods are not refined enough to be used as a key for encryption, and will require a backup authentication method to decrypt a file system, as with pattern and fingerprint unlocks on Android. Others can even authenticate in certain situations but use a backup otherwise, like how a location based authentication will unlock the phone automatically if the user is in a known location, but might switch to a different authentication method in an unknown location. Sadly, many of these compound methods are not particularly popular among users, even if these options are readily available and easily accessible [16].

1.3.1 The Concept of Trust

A more modern approach to authorization has been proposed in recent years, with the suggestion of a “trust score” to a user, where if a user can’t be completely authenticated, they are only authorized to do a limited range of things on the device [17]. For example, if a phone has a paired Bluetooth smartwatch that has a weak connection to the phone at the moment, it could trust that the user was near enough to the phone, but not completely certain that the user was the one using the device due to the distance of the watch implied by a weaker signal. In that case, the phone could require no further authentication for simple, general tasks like using a web browser, but would require additional authentication for using a banking app. Trust scores are an excellent application for our use, as it allowed us to compare the likelihood of a user performing along their usual behaviors against their current ones, available well enough to authenticate users on a spectrum as opposed to the two options of authenticated or unauthenticated.

1.4. Limitations of Current Security Methods

Most password systems are not secure enough to protect the data that a smartphone holds, since they tend to trade off major security for convenience. Table 2 describes some of the major vulnerabilities of the above methods.

Method	Description
Pin	Generally short and can be easily stolen by an attacker observing the phone being unlocked. One study suggests that 15% of 4 digit pin lockscreens can be unlocked in just ten tries [18]. Longer pins may be an option but they are generally easily forgettable.
Password	Password database leaks are far too common among large corporations holding them, and many people use the same password in multiple places, potentially compromising their accounts across several services with a compromise at the least secure of all the companies holding them.
Pattern	Pattern lock screens are vulnerable to smudge attacks, where the oil residues of fingers make it possible for people or computers to trace the pattern and gain access [11].
Pictures	Also vulnerable to smudge attacks with swiping and drawing. It is also relatively easy to find the parts of a picture that one may touch.
Fingerprint	Fingerprint passwords are unchangeable and they are irreplaceable. It is also possible to fake fingerprint data by gathering information through photographs of the target's hands [19]. Government fingerprint databases have also been breached [20].
Iris	Iris passwords are unchangeable and irreplaceable. They have similar vulnerabilities to fingerprints.
Trusted Devices	An attacker could take the target's phone and unlock it if they were near the target and their trusted device.

Voice Recognition	Accuracy may lower if the user's voice changes due to certain conditions such as being sick [21]. Passphrases can also be easily overheard.
NFC Tags	NFC tags can be stolen in the same way a smartphone can.
Location-Based	An attacker just needs to unlock the target's phone in a specified location to gain access

Table 2. Vulnerabilities of Current Smartphone Authentication Methods

Many users tend to utilize security methods that are more convenient and easier to use. For many people, convenience is often more important than security, and once a hacker has gained access, none of these methods stop an attacker from stealing data from the smartphone after it has been unlocked. It becomes trivial for such an attacker to steal important information from a target's unlocked smartphone. In general, users tend to remain logged in to multiple accounts on their phones. Attackers can easily access the target user's social media accounts, email accounts, and other services.

1.5 Behavioral Authentication

A notable new direction to assuage this problem is to take the limitations of explicit authentication and approach authentication through a new direction- user behavior. Behavioral authentication may solve many of the shortcomings of current smartphone authentication techniques. Conceptually, a behavioral authentication system can learn who a user is by understanding how they act and interact with their phone and requires no additional input from the user to authenticate him or her. In current techniques, an attacker would just need to acquire certain information to get into the system. Unlike current authentication techniques, it would be very difficult for an attacker to emulate a victim user exactly even if they were able to acquire

information on the victim [2]. Because behavior is implicit, it can be checked continuously without being an annoyance to the user as opposed to just when unlocking the phone [22].

1.6 The Goal of This Project

We aim to enhance smartphone security by creating the beginnings of a low power, customizable, continuous behavioral model system (BMS)- a security system that authenticates users based on a variety of behavior patterns and unique phone interactions. Conversely, this system will also be able to de-authenticate a user, when it detects inconsistent or suspicious behavior. We envision that BMS:

- Tracks the user's behavior through a combination of metrics selected for their ability to capture unique user behaviors around smartphone usage
- Transmits the tracked data to a centralized server for storing data and training a model of the user using deep learning
- Compares new behavioral user data against a trained model of the user and returns whether the user is authenticated or unauthenticated based on how well new behavior matches the known behavior

The proposed system uses behavioral data gathered from the phone's sensors to solve some of the problems associated with current password systems, namely the attacker stealing an unlocked phone from the actual user. Because the system continuously authenticates based on behavior, critical applications or the phone can be locked out if some of the tracked features change dramatically yielding a low trust score.

One major benefit of the proposed system is that it is convenient and easy to use. It can be used in combination with other authentication methods and it does not require any additional steps from the user except to use their smartphone and behave like they normally do. However, a system that continuously runs in the background inherently has a higher energy cost than an episodic system. Regardless of the convenience of authentication and enhanced security, most people have an interest in maximizing the battery life in their smartphones and it would be a key concern for us to minimize battery drain [7].

This system is also modular. The system can track as many behaviors as necessary to reliably authenticate the user. To keep the project's scope manageable, BMS will initially be trained to only track a small number of specific behaviors. However, as more behaviors that can be used for authentication are discovered, and evaluated, the proposed system can be modified to integrate them. Comparatively, BMS is more flexible than other password-based authentication systems.

2. Background

2.1 Data Smartphones Collect

Smartphones have sensors collect usage data in several different forms. Usage logs amass information quickly, as phones collect data from applications, services, sensors, and more [23]. These logs can contain simple data such as SMS and call history, but also more complex information such as location, Bluetooth, and Wi-Fi statistics. Although logs only generally contain the data that phones can passively collect, it is possible to actively obtain more data by utilizing the many different sensors smartphones contain and running services to collect such data.

2.1.1 Hard Sensors

Hard sensors are "physically-sensing devices" that sense environmental (e.g. light, temperature) or physical behaviors (e.g. motion, touch) of the user [24]. These collect data from smartphone hardware such as the accelerometer, GPS, and gyroscope. The following (Table 3) is a list of common smartphone hard sensors and what data they can collect [25].

Hard Sensor	What it Measures
Accelerometer	The acceleration force that is applied to a device in the x, y, and z axes.
Gyroscope	The phone's rate of rotation about the x, y, and z axes.

Light	The ambient light surrounding the device.
Magnetometer	The Earth's (and other) magnetic fields in the x, y, and z axes.
Orientation	The degrees of rotation that a device has made about all three axes.
Barometer	Measures the ambient air pressure around the device.
Proximity	Measures the proximity of an object relative to the distance of the screen.
Pedometer	Measures the number of steps taken by the user.
Touch	Measures length and width of strokes by the user on the touch screen.
Temperature	Measures room temperature
Humidity	Measures relative ambient humidity

Table 3. Common Smartphone Hard Sensors[25]

2.1.2 Soft Sensors

Soft sensors are sensors that "record information of a phone's running status" [24]. These collect information from a phone's software. Some examples of soft sensors include screen on/off, app usage, messages, and phone calls. The following (Table 4) is a list of smartphone soft sensors analyzed by Rachuri et al [26].

Soft Sensor Features
Battery charging? (boolean)
Battery level {1,2,...100}
Battery state {low, medium, high}
Network type {Wi-Fi, cellular, none}
Network name (string)
Last app used category {app categories}
Proximity events

Screen events
SMS events
Phone calls
Recent SMS/Calls? (boolean)

Table 4. Soft Sensor Features used by Rachuri et al [26].

2.1.3 Inferring User Activity

Once sensor data is accumulated, smartphones can take the information saved in these logs and use them across applications that request sensor data. Using the data from these sensors and logs, it is possible for the phone to make inferences about the activities of a person. There are many examples in the literature of smartphone applications that detect when the user is awake and when they are sleeping, the walking gait of a user, and the mood of the user by using a combination of these sensors, which are described further below.

Smartphone Inference of Alcohol Consumption Levels from Gait tracks and classifies smartphone accelerometer data to infer the user's level of intoxication [27]. *Toss 'N' Turn: Smartphone as Sleep and Sleep Quality Detector* analyzes accelerometer, microphone, ambient light, screen proximity, running process, battery state, and display screen state data to create models that predict when the user is sleeping and sleep quality [28].

2.2 Doze: Android's Power Saving Mechanism

In order to reduce the energy consumption of the Android operating system has various energy-saving mechanisms and ways of interacting with data in real time. Specifically, it's common for when and how smartphone data can be used to be limited. These limitations have been developed over time through improvements to Android's power-saving functionality, to allow the phone to last throughout the day and for extend periods of non-use. Such

implementations are comprehensively referred to as ‘Doze’, since Android Marshmallow (6.0), and Doze On-the-Go, implemented since Android Nougat (7.0) [29]. Since Doze limits the frequency with which data can be retrieved on the client and sent to the BMS server in order to verify the user’s behavior as consistent and authentic, it is vital to understand how the Doze mechanism functions.

2.2.1 Doze in Android 6.0 (Marshmallow)

Doze, a feature of modern versions of Android, forces wakelocks to be in sync with one another and less frequent. With Doze, the device needs to be in a relatively stationary position, like sitting in a nightstand or pocket at any given time to be enabled. When Doze is on, low-priority apps, such as email notifications, are relegated to infrequent and low-power activity: they are forced to wait for their notifications, background updates, and more until the phone is active again or on a time scheduled by Doze [30]. Only high-priority applications, like apps that make and receive calls, send and receive text messages, and other apps flagged as high-priority through the Google Cloud Messaging service, can prompt a wakelock. As seen in Figure 1 below, apps are not allowed to access data except for the normal ‘maintenance windows’ available.

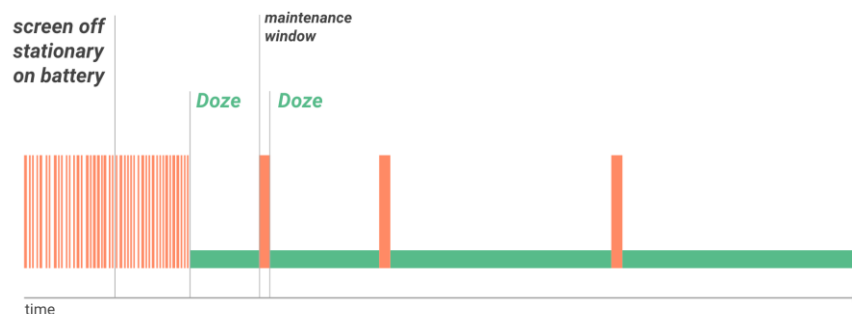


Figure 1. Wakelocks in Android with Doze (off, then turned on) [31].

2.2.2 Doze's Expansion in Android 7.0 (Nougat)

In Android 7.0 (Nougat), a version of Android several of our development phones used for this project run, introduced even more considerations from Doze for development of our application. Doze was expanded with Google's "Project Svelte", where Doze limits broadcasts for mobile networks and Wi-Fi; apps will frequently ask for data updates with a `CONNECTIVITY_ACTION` change, and instead of them broadcasting for this when a connection is changed, app permissions are changed to the point to remove this, and also keep devices unable to receive information regarding a new picture or new video.

2.2.3 Implications of the Doze Mechanism and Recent Changes

With these major Doze-related changes to how Android allows for scheduling and accessing data, our implementation is constrained to be less aggressive in triggering data collection events, as we have to wait for Android's job scheduler to select an acceptable time for the BMS client to obtain data, rather than be collecting data on demand, like we had initially hoped for. We will instead have to consider how frequently our smartphone client can poll sensor data, with the OS allowing our app to wake up and ask for high priority data. Additionally, it significantly limits how often we can communicate with a server, as wakelocks to send data are limited to the 'maintenance windows' that the operating system allows our application. With these considerations in place, the design of our project is different from that of many similar, past implementations of behavioral authentication, as we need to work around the restrictions Doze and other optimizations pose while maintaining a strong security model. We are limited by

the frequency at which the user's smartphone can send or receive data from our authentication server.

2.3. Deep Learning/Neural Networks

In implementing an intelligence that can log activities and convert these gathered metrics to a conceptually understood behavior, we found Artificial Neural Networks (ANN) to be our most viable option. ANNs are information processing systems that can be used to detect patterns in data and sort data. ANNs are loosely modeled after the interconnected neurons in a mammalian brain structure [32]. Generally organized into 3 types of layers, ANNs include an input layer, one or more hidden layers, and an output layer. Each of these layers are made up of interconnected nodes which contain an activation function to fire. Generally, information or patterns are given to the input layer, the input layer then communicates to the hidden layer(s) to process the information, and the last hidden layer communicates to the output layer which then displays a result. An example neural network is shown in Figure 2 below.

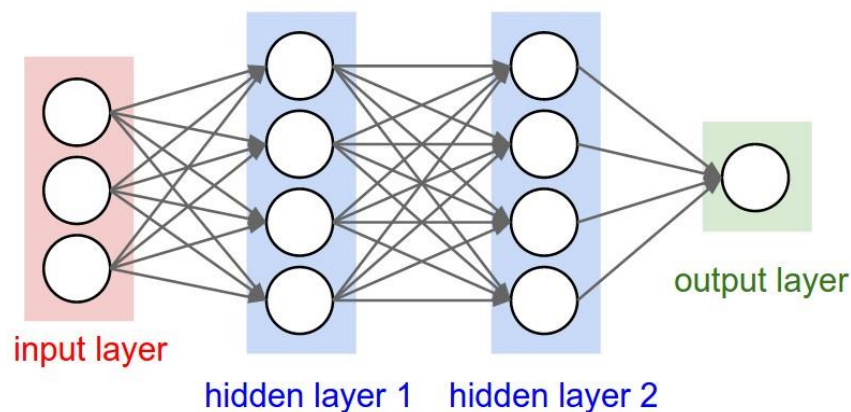


Figure 2. An example neural network [33]

ANN's can vary wildly, with some neural networks better at some things than others.

Below, Table 5 lists some types of neural networks and their uses, as described by The Asimov Institute [34].

Neural Network Type	Uses
Feed forward	Supervised learning, straight-forward correlations between input nodes.
Convolutional	Image classification, filtering image detail.
Recurrent	Processing events in a timeline, data given in a sequence.
Support vector machines	Classifying n-dimensional data.
Kohonen networks	Competitive learning to classify unsupervised data.

Table 5. Some Common Neural Network Types [34]

We used a feed forward neural network because of its ease of implementation and its ability to find correlations between the input and output. In training these distinct networks, there are two main approaches: supervised and unsupervised learning. Supervised learning trains an ANN by giving it inputs as well as the correct output. If the system's output is different from the desired output, the system will adjust its structure in order to better categorize results in the future a process to adjust itself so that it will be better at categorizing in the future. Unsupervised learning is when a network is given many unlabeled inputs and makes categories and classifications by looking at patterns in the data, no corrections are made since there is no desired output. After training, an ANN can process data and classify it based on the weights and biases of nodes learned during the training session. We plan to use a supervised approach since our training data will be labeled with the correct outputs. For instance, data collected from subject A can be labeled as "true" for subject A and "false" for all other subjects.

Incremental learning is a technique used in supervised machine learning wherein the neural network is trained with each data point exactly once. It can be used in situations where

the data is generated or becomes gradually available over time, or when the size of the training data is too large to maintain all of it in a single repository. In our case of continuous authentication, data is being collected non-stop throughout the day. Since our authentication system must continuously incorporate this new data, we plan to use incremental learning. Further, with each data point used in training exactly once, not after the model has been used to predict and later train, data can be deleted immediately after training to simplify memory management.

3. Related Work

Smartphones contain many sensors that can be used to track behaviors of the user. Many researchers have purposed these sensors in determining which combinations of behaviors positively identify a user, and evaluating the effectiveness of these sensors for that task. We have used these studies, discussed below, to build an understanding of prior work in the field. We drew inspiration from the behaviors that were tracked, prior findings regarding what behaviors were most accurate and robust: from the human studies conducted, the artificial intelligence models used, and their goals for future work upon conclusion of their research. The area of behavioral authentication is actively being researched and highly collaborative- many groups utilize one another's studies to improve their own research, and make a more significant contribution to the field. Due to our limited time, we utilized the findings and results from other studies to make decisions such as sensor selection, so that we can focus on other, less-researched aspects.

3.1 Comparative Research on Positive Identification

Positive Identification, or research based on accurately determining that the user using the device is one that is authorized to work with it, is by far more difficult compared to negative authentication, which only has to have a stronger confidence that the identification is correct or not close enough to what is expected. In Lee et al. [24] they attempt to positively identify and distinguish users based on three sensors. They chose their three - magnetometer, accelerometer, and orientation, based on the diverse coverage of user and environmental information, along with the lack of user permissions required to track these sensors [24]. They acquired accuracies for unique combinations of the three sensors, and measured tradeoff between training time and

sampling rate. Further, they provided a comparison between their study and similar studies involving smartphone sensors; we examined this research compiled by Lee et al. in order to get a better understanding of metrics that worked for other groups [24]. Table 6 is a table of this research. We built off of these studies and, from the sensors listed, we incorporated two of these sensors, orientation and GPS. two of these sensors, orientation and GPS.

	Devices	Sensors	Method	Accuracy	Detecting Time	Script
Lee et al.	Nexus 5, Android	Orientation, Magnetometer, Accelerometer	SVM	90.23%	Train: 6.07s Test: 20s	No
Kayacik et al., 2014	Android	Light, Orientation, Magnetometer, Accelerometer	Temporal & Spatial model	N/A	Train: N/A Test: $\geq 122s$	No
Zhu et al. 2013	Nexus S	Orientation, Magnetometer, Accelerometer	N-gram language model	71.3%	N/A	Yes
Buthpitiya et. al., 2011	N/A	GPS	N-gram model on location	86.6%	Train: N/A Test: $\geq 122s$	No
Trojahn et al., 2013	HTC Desire	screen	Keystroke & handwriting	FP:11% FN:16%	N/A	Yes
Li et al., 2013	Motorola Droid	screen	Sliding pattern	95.7%	Train: N/A Test: .648s	Yes
Nickel et al. 2012	Motorola Milestone	accelerometer	K-NN	FP:3.97% FN:	Train: 1.5min Test:30s	Yes

				22.22%		
--	--	--	--	--------	--	--

Table 6. A Comparison of Sensors, Methods, and Accuracy for Identification/Authentication[24]

Another study, by Yampolskiy et al[35]. researched other behavioral metrics studies and showcased how single metrics were able to achieve a high detection rate while having a low false acceptance rate and false rejection rate. Below, Table 7 shows the various behavioral metrics which Yampolskiy compared. Although we do not be using the exact metrics they mentioned, their research has led us to believe that metrics such as App usage events will work well.

Behavioral metric	Publication	Detection Rate	FAR (Acceptance Rate)	FRR (False Rejection Rate)	EER (Equal Error Rate)
Biometric Sketch	Bromme and Al-Zubi (2003)				7.2%
Blinking	Westeyn and Starner (2004)	82.02%			
Calling behavior	Fawcett and Provost (1997)	92.5%			
Car driving style	Erdogan et al. (2005a)	88.25%			4.0%
Command line lexicon	Marin et. al. (2001)	74.4%		33.5%	
Credit card use	Brause et al. (1999)	99.995%		20%	
E-mail behavior	Vel et al. (2001)	90.5%			
Gait/stride	Kale et al.	90%			

	(2004)				
Game strategy	Yampolsky and Govindajaru (2007)				7.0%
Handgrip	Veldhuis et al. (2004)				1.8%
Haptic	Orozeo et al. (2006)		25%		22.3%
Keystroke	Bergadano et al. (2002)		0.01%	4%	
Lip movement	Mok et al. (2004)				2.17%
Mouse dynamics	Pusara and Brodley (2004)		0.43%	1.75%	
Programming Style	Frantzeskou et al. (2004)	73%			
Signature Handwriting (1)	Jain et al. (2002)		1.6%	2.8%	
Signature Handwriting (2)	Zhu et al. (2000)	95.7%			
Tapping	Henderson et al. (2001)				2.3%
Text Authorship	Halteren (2004)		0.2%	0.0%	
Voice/speech /singing	Colombi et al. (1996)				0.28%
Voice/speech /singing	Tsai and Wang (2006)				29.6%

Table 7. Recognition, verification, and error rates of behavioral metrics

3.2 Generalized Algorithm for Categorizing Behavioral Authentication

Yampolskiy *et al.*[35] organizes behavioral biometrics into several categories. According to those authors, there are 5 types of Behavioral Biometrics. They are described in detail below:

- **“Authorship based biometrics”**: Metrics, for example, that examine a piece of text or a drawing. For authentication, these algorithms look at the style peculiarities such as vocabulary, punctuation, and brush strokes.
- **“Human Computer Interaction biometrics”**: Metrics that consider human interaction with input devices; they are indicative of the specific skills, styles, and knowledge displayed while interacting with a computer.
- **“Low level computer software action”**: Metrics that read from system logs and activities indirectly generated by the second category.
- **“Motor-skills for verification”**: Metrics that look at muscle movements while performing tasks.
- **“Pure Behavioral biometrics”**: Metrics that revolve around strategies, skills, and knowledge shown during the performance of mentally demanding tasks, with no body measurements involved

The authors developed a generalized algorithm for these different behaviors, which follows as:

1. Pick a user behavior.
2. Break-up the behavior into component actions.
3. Determine the *frequencies* of component actions for each user.
4. Combine results into a feature vector profile.
5. Apply similarity measure function to compare the stored template with current behavior.

6. Experimentally determine a threshold value within which the user is authenticated.
7. Verify or reject the user based on the similarity score comparison to the threshold value.

The authors suggest that comparison and analysis of each feature follow guidelines of universality, uniqueness among individuals, the ability to adjust user behavior of time, and the ease of collectability. Furthermore, they made note that only some behavioral biometrics are dependable enough to be usable for any level of identification. In one case, the scores for a standardized test, such as IQ test, SAT, GRE, or GMAT, are not enough to identify an individual, because of a lack in uniqueness of the scores, but they could have been combined with other biometrics to improve accuracy. By transitioning these individual studies into a generalized approach, new behaviors proved much easier to be analyzed, tested for viability, and incorporated in our system. In all, these concepts helped us better research behaviors before we tested them, along with helping us disregard behaviors that are not particularly helpful with authentication.

3.3 Deep Neural Nets for Modeling Mobile Soft Keystroke Authentication

Deng and Zhong present a deep learning approach for mobile keystroke dynamics biometrics, as well as exploring the additional sensory data (touch screen, accelerometer, and gyroscope) available on smartphones for augmented keystroke biometrics. After describing previous works in the area, the authors suggested using Deep Neural Network (DNN) modeling on mobile soft keystroke authentication. The authors used the timing (duration of touch), tapping (size of touch area), and inertial sensor (accelerometer) features, combined with the Stanford TapDynamics mobile keystroke dataset to train their DNN. The dataset included key tap, latency,

and accelerometer data from 55 subjects typing a pin number into a phone, yielding a total of 1704 data samples recorded and 35 generated features. The authors performed layer-wise unsupervised training with the data to pre-train a Restricted Boltzmann Machine, a network with a visible and a hidden layer. Afterwards, they tuned the neural network with stochastic gradient training; the system had a False Acceptance Rate (FAR) of 4.4% and a False Rejection Rate (FRR) of 5.3% when no features were removed, an 11.7% FAR and a 12.6% FRR with just accelerometer statistics, a 17.8% FAR and 14.7% FRR with just key tap sizes, and a 28.4% FAR and 17.4% FRR with just key tap duration. These FARs and FRRs showcase Neural Networks effectively used to analyze behavioral data that then authenticates users. Although there exists a lack of research in Neural Networks implemented as a method of analyzing data, this study demonstrates Neural Networks' potential in classifying data, further justifying our decision for using them to model user behavioral data.

3.4 Touchalytics

Frank *et al.*[2] obtained high accuracy authentication by classifying 30 touch data features, listed in Figure 4. Participants went through an “enrollment phase” (modeling phase), during which horizontal and vertical swipes were tracked. For the purpose of obtaining natural test results, participants were asked to read three documents and answer comprehension questions on a custom app that logged touches [2]. The unique touches of eight different users that were logged during this study can be seen in Figure 3. The authors propose a classification framework using k-nearest neighbors algorithm and Support Vector Machines to learn user behavior. During the continuous authentication phase, the participant must re-authenticate

after a certain number of consecutive negative classification results. In our implementation, we used a similar enrollment phase to train our model, followed by a continuous authentication phase to test it. An “informative” value was calculated for each of the 30 touch features in the Touchalytics research to rank each in terms of its ability to identify the user, as seen in Figure 4. The ranking gives us an approach for appropriately weighting these touch features during our initial implementations where we use a static metric ranking. After the model is trained, we expect the dynamic weights, obtained through training our Neural Network, to generally reflect this data.



Figure 3. Smartphone strokes recorded for eight different users, showing unique user touch behavior

Rel. mutual information	Feature description
20.58%	mid-stroke area covered
19.63%	20%-perc. pairwise velocity
17.28%	mid-stroke pressure
11.06%	direction of end-to-end line
10.32%	stop x
10.15%	start x
9.45%	average direction
9.43%	start y
8.84%	average velocity
8.61%	stop y
8.5%	stroke duration
8.27%	direct end-to-end distance
8.16%	length of trajectory
7.85%	80%-perc. pairwise velocity
7.24%	median velocity at last 3 pts
7.22%	50%-perc. pairwise velocity
7.07%	20%-perc. pairwise acc
6.29%	ratio end-to-end dist and length of trajectory
6.08%	largest deviation from end-to-end line
5.96%	80%-perc. pairwise acc
5.82%	mean resultant length
5.42%	median acceleration at first 5 points
5.39%	50%-perc. dev. from end-to-end line
5.3%	inter-stroke time
5.14%	80%-perc. dev. from end-to-end line
5.04%	20%-perc. dev. from end-to-end line
5.04%	50%-perc. pairwise acc
3.44%	phone orientation
3.08%	mid-stroke finger orientation
0.97%	up/down/left/right flag
0%	change of finger orientation

Figure 4. Features' effectiveness to identify an individual.

3.5 Soft Authentication with Low-Cost Signatures

Buthpitiya *et al.*[36] analyzes data from multiple (low accuracy) sensors to create an accurate authentication system. Their approach uses low-power cost strategies, such as reading logs that are already updated during routine usage. Utilizing these very practical measurements, the behavioral metrics analyzed included message response patterns, calling patterns, outdoor mobility patterns, and indoor mobility patterns. Compared to other research on behavioral security in smartphones, this paper focuses heavily on the *phone* uses, such as calls and text messages, addressing the original uses of the smartphone. These aspects prove important

considering their uniqueness to smartphone use rather than tablets or other computers that cannot make or receive calls. This article investigates a variety of feature and extraction modeling approaches; of particular interest are the n-gram model and Hamming distance models for GSM and Wi-Fi signal strength, respectively. We will likely implement similar strategies for analyzing user location. The Hamming distance is important in that it can quantify a distance between a set of non-numerical labels, which can be used to compare lists of Wi-Fi networks, Bluetooth devices, used apps, or locations. Overall, the n-gram model can be used to find sequential patterns across data points, as opposed to patterns within a data point [37].

3.6 Continuous Authentication on Mobile Devices Using Power Consumption, Touch Gestures and Physical Movement of Users

Murmura *et al.* presented an approach of authenticating users through a combination of touch gestures, power consumption, and physical movement, while also taking into account application context when modeling user behavior [38]. They collected data from 73 regular and irregular smartphone users in a room using Google Chrome and Facebook on a Nexus 5 Android phone. The phone logged power consumption; the touch area, pressure, and coordinates of finger touches; the number of fingers touching the screen; and short-term physical movements, such as hand vibrations. The authors noted that noise was a significant factor in the data, even though their subjects were in a highly controlled environment. However, they were able to authenticate users with a 93% accuracy after collecting data by additionally modeling the present noise. The authors believe that their approach is more viable as a real-world solution than other competing approaches in literature. Specifically, it can be scaled to a more realistic scenario, since noise is

already accounted for and a mobile device could be used over multiple days by volunteers performing daily routine tasks. We need to ensure that we test our metrics in both controlled and uncontrolled environments-- the point of behavioral authentication is to use personal uniqueness to identify and authenticate people, and using neural networks, we can create models that fit individuals.

3.7 Authentication Feature and Model Selection using Penalty Algorithms

Another small study conducted by Murmura and Angelos [39] suggests that the best authentication features for finger swipes include:

- Arc-length of gesture
- Direction between endpoints
- Average finger diameter during gesture
- Average pressure during gesture
- Average finger speed during gesture

They also suggest that the best authentication features for taps include duration of gesture, direction between end-points, average finger diameter during gesture. The authors collected WhatsApp data from 110 users' datasets, with each user having at least 350 swipes and 350 taps each. 40 of the users had over 2 hours of usage logged on the application. They used the sum of Euclidean distances to an observation's k-nearest neighbors to get a measure of uniqueness. We can use these findings into our system when we try to incorporate touch gestures into our application.

3.8 Miscellaneous

Ballard *et al.* [40] discusses the impracticality of testing behavioral security tests in “lab” conditions as many other researchers have done. The paper shows that security methods are much weaker than they appear when they are tested against “trained” forgers. The participants gave handwriting samples and then submitted forgeries of other participants’ samples. The best 9 out of 55, the “natural” forgers, were allowed two hours of training to practice forging handwriting samples. The equal error rates were found to be exaggerated by 375% if the “trained” forgers’ analysis was used instead of the naive analysis. The forgers’ improvement was graphed over time, showing that training had a significant effect on successful forging, even for the “natural” forgers selected. Additionally, artificial intelligence was used to learn from handwriting samples and attempt to bypass the security. In order to improve the robustness of our authentication method, we will also have participants attempt to break our security. We will have some participants attempt to emulate other participants and see if they are able to break our security measures. An individual’s model will be trained with both his/her training data and against the training data of others. In addition, the model will be tested by having participants attempt to imitate the behavior of other participants.

3.9 Summary

We devised a list of requirements for a behavioral model system by analyzing the papers that we have reviewed. These include:

1. Accuracy
2. Power efficiency
3. Ease of implementation

Many of these studies utilized different combinations of metrics. Several of these used the inbuilt GPS, touch screen, and phone orientation with varying degrees of success. A basic requirement of our study is choosing metrics that have been proven to be somewhat effective for authentication by other papers, such as GPS, and pairing them with others that are not as proven, such as app usage. App usage falls under the category of a low accuracy power efficient sensor, which provide a better overall picture of phone usage in addition to the more accurate sensors. Touch is one biometric that is a unique enough identifier on its own, and would be included if not for the difficult implementation. Overall, a good behavioral authentication system depends on carefully selecting a combination of behavioral metrics that work together to build a suitable representation of one's behavior.

4. Methodology

To be able to authenticate a user based on their behavior alone is a challenging task, and more so distinguishing individuals who exhibit similar behaviors based on the metrics we used. To understand the data that phones accumulate, we used four metrics we found to be effective at authenticating users, as suggested in previous and related works.

4.1 Pilot Study

We chose to test our system by having 4 senior WPI computer science students install our BMS data collection application and consent to having their phone statistics continuously monitored. They were asked to use their smartphones normally and sign into their Google Accounts to have their phones automatically upload their data. They were then asked to share their data folder with our BMS Google Account. Upon having test users collect data sorted by time of day, date, and device, we eventually had the data accumulated across multiple devices and added to a database on a personal development server of ours. We fed the data we collected into a neural network and had the system learn what is acceptable behavior independently and created models of test subject's behaviors. We then tested the test subject's data against each other's models to check the strength of the models.

4.2 Assumptions

We made the following assumptions during the testing period:

1. The user's behavior will not be affected by the use of our application.
2. Each user's smartphone will be used only by the user and no one else.

3. Conversely, each user will use only one smartphone.
4. Subjects are relatively distinguishable.
5. The user's everyday and weekly routines do not change significantly during the app's use.

4.3 Metrics Selected for Inclusion in BMS

We implemented our implicit authentication by selecting four behavioral metrics we found to be most promising from our literature review in identifying a user uniquely. In our research, we reasoned that it was most important to be able to try unique and distinct sets of behaviors especially in the beginning of our development, so we could cover a large subset of the meaningful data collected by the phone as used for authentication. The four metrics we chose were app usage data, precise GPS location, Wi-Fi signal strength, and tilt/angle from the accelerometer. These metrics are discussed in detail below.

4.3.1 Smartphone App Usage Events

App usage events are events that log how a user interacts with their phone throughout the timeline of their daily activities, by logging when apps are moved to the foreground (actively open and used) and background (minimized) on their device. App usage proves a promising metric with the nature this data in relation to the behavior of the user [41]. Like how screen captures on a desktop computer allow an observer to see actions being undertaken by users, understanding what apps are being used at any given point in time enabled us to develop a behavioral model towards their app usage. By monitoring the times of day that particular apps are used, we understood when a user typically finds this app useful, and if there are daily, weekly, monthly, or other patterns to this app's usage. Inserting the concepts of app usage frequency, or

how many times the app is opened over a day or other timespan, and app usage duration, or how long the app remains in the foreground per each usage event, we began to develop a timeline of user interactions with their phone. This intuitive and natural formation of a model of how the phone is used on different timescales was made easier by the approachable nature of the APIs that gather this data already for us, but do nothing with it. Fridman *et al* included app name and frequency in their active authentication system and found that across three different time windows of 1 minute, 10 minutes, and 30 minutes, app usage was the least predictable contributor overall compared to text, web, and location [42]. However, for the smaller windows the contribution of app usage was higher, which can be explained by the first app opened in a window being a consistent behavior [42]. By amassing this data into timelines across varying scales of how the person usually behaves in regards to their phone, we gave the phone a basis for what behavior is to be expected out of it on a normal day. Furthermore, more precise metrics can be observed through app usage, including battery drain from an app over a frame of time (someone using the Facebook app scrolling through their news feed would typically use a lot less power than someone who frequently watches videos on the app, for example), along with app transitions to form a path which users tend to follow between apps.

4.3.2 Precise Location

The location metric includes the longitude and latitude values from the GPS. To better provide a state to the model, we found it imperative for states we measured to have an associated location, or a precise latitude and longitude measured by GPS. With location already used in Google's 'Trusted Locations' for authenticating Android devices when they are within the range of a location specified, we realized that it was an invaluable metric to couple with others to give

us a larger picture of the state of the user [43]. Recognizing that we could tether location to app usage metrics, we understood not only when a user interacts with a device and with what app, but also where the user is at the same time. Upon these more complete associations, we used pattern-based location history as a metric to map users' behavior against, to see if they returned to locations they visited previously, giving us more points of comparison for behavior of the user. Location, and precise location at that, allowed us to gain a completely new level of precision for combinatorial behavior, or the combinations of all the differing behaviors that we measured; it added millions of potential points for users to be located at that are unique to them, or unique in combination with other metrics the phone observed. Additionally, with location behavior being incredibly personal, we were explicitly clear with how we handled the data we gathered, how it was stored, and how long it was used for app authentication purposes.

4.3.3 Wi-Fi Access Points Observed

Wi-Fi provides smartphones with internet access at home and on the go wirelessly. The list of Wi-Fi Access Points (APs) a user observes per every collection cycle offer a deeper layer of context for the user's location and travel path, which precise location cannot provide. In a common scenario where many people are all living in the same location they will all show similar behaviors. Using signal strengths to approximate distances from access points, the Wi-Fi APs around a user provide a finer level of detail that can provide the position of a user within a building. Wi-Fi Information is already gathered by the phone periodically. As a result, our system's usage of this metric is opportunistic and the power cost of this metric is thus minimal.

4.3.4 Tilt Data

Finally, to complement the other three metrics, we associated accelerometer and magnetometer data to instantaneously calculate the three-dimensional tilt of the phone. Since tilt can associate how the user interacts with their phones, we could see the orientation of the phone while using different applications, at different locations or in different states, such as surfing web pages, browsing photos, typing, and much more. People habitually interact with their phones in ways that are comfortable to them, and tilt gave us an idea of how the user holds their phone across different apps.

4.4 Gathering the Data

We found our application for data gathering to be best logically structured as a stream, or a timeline of data gathered across the day. The flow of data is shown in Figure. 5 below. Each sensor is scheduled to record data at regular intervals. When data is pulled from a sensor, our Behavioral Model System application saves the recent sensor activity and a timestamp to a file on the device. Then file is uploaded regularly to a server for deep learning processing. Finally, the server will send a message back to the phone showing the authentication status.

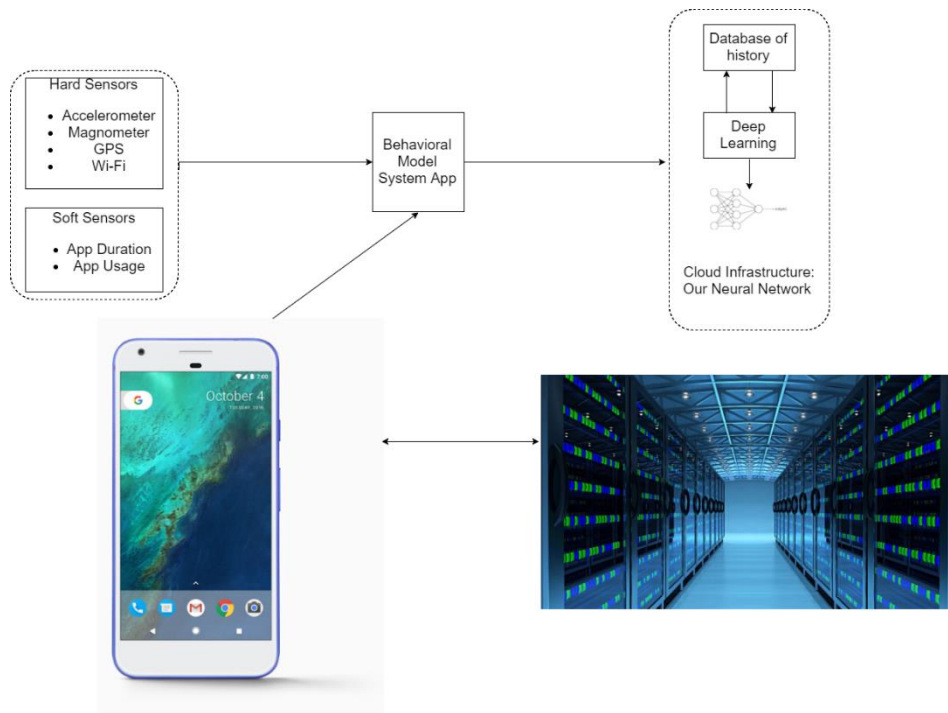


Figure 5. Feeding into the ANN for an authentication output

4.5 Privacy Concerns

Because we are collecting and analyzing user data, we needed to address privacy concerns with user identity and tracking their behaviors. We chose to address this problem in two ways. We used anonymization techniques in both the way we stored files on our server and in the way that we structured the user data. The files for each individual user used the same formatting and contained no identifying marks that pertain to name and phone number of the user. We also allowed users to disable metrics that they did not want to be used for authentication purposes.

4.6 Authentication

The user was authenticated or unauthenticated based on a trust score. A trust score represents the likelihood of the user matching their trusted model from a range of zero (untrusted) to one (trusted). As a trust score gets closer to one, the significance of a single trust

score being higher than another trust score decreases. If the trust score was below a certain threshold, the user would be de-authenticated. The threshold for de-authentication was determined experimentally. In considering trust, the cutoffs cannot be so strict that the user is often and inconveniently de-authenticated or so lenient that significant damage may be done before an imposter is unauthenticated. Through the accumulation of more data however, the user's trust score became more accurate over time, while remaining dynamic and consistent for only one particular user's threshold range.

We used a feedforward neural network with several input nodes for each sensor and input nodes for the timestamp at which the data point was recorded. For data such as app usage metrics, which has a potentially infinite list of used apps, we processed each one of the apps independently. There was one output node, producing the trust score. The number of hidden layers were determined experimentally. In order for the authentication to be continuous and convenient, as we intended, it had to: 1) adjust to new changes in the person's routine 2) authenticate accurately 3) weight features effectively and 4) be practical for a diverse group of users.

The neural network constantly evolved as its training data changed. This was done with incremental learning, training the neural network with just one input at a time. After each training session of size one, the neural network updated its weights and biases in a direction that best accommodated the one data point. Furthermore, the learning rate parameter was adjusted to modify how sensitive the neural network is to change.

5. BMS System Implementation

There were several important considerations we followed for our implementation of the Behavioral Model System, or BMS. Accuracy was one of those considerations because a user needs to be authenticated and unauthenticated correctly for the system to work. Aside from accuracy, another significant consideration for this project was extensibility- we made it possible to add and remove a near indefinite number of metrics at any given time, and have our program adapt.

Figure 6 shows the overall design of our system. Sensors on the phone are tracked through our Behavioral Model System App. The sensor data is saved as a JSON file using a JSON saving Daemon and then uploaded to Google Drive cloud storage. The cloud storage moves the data to a cloud infrastructure which processes the data using a neural network and produces an authentication result. The result is sent back to Google Drive and then to our app's authentication service.

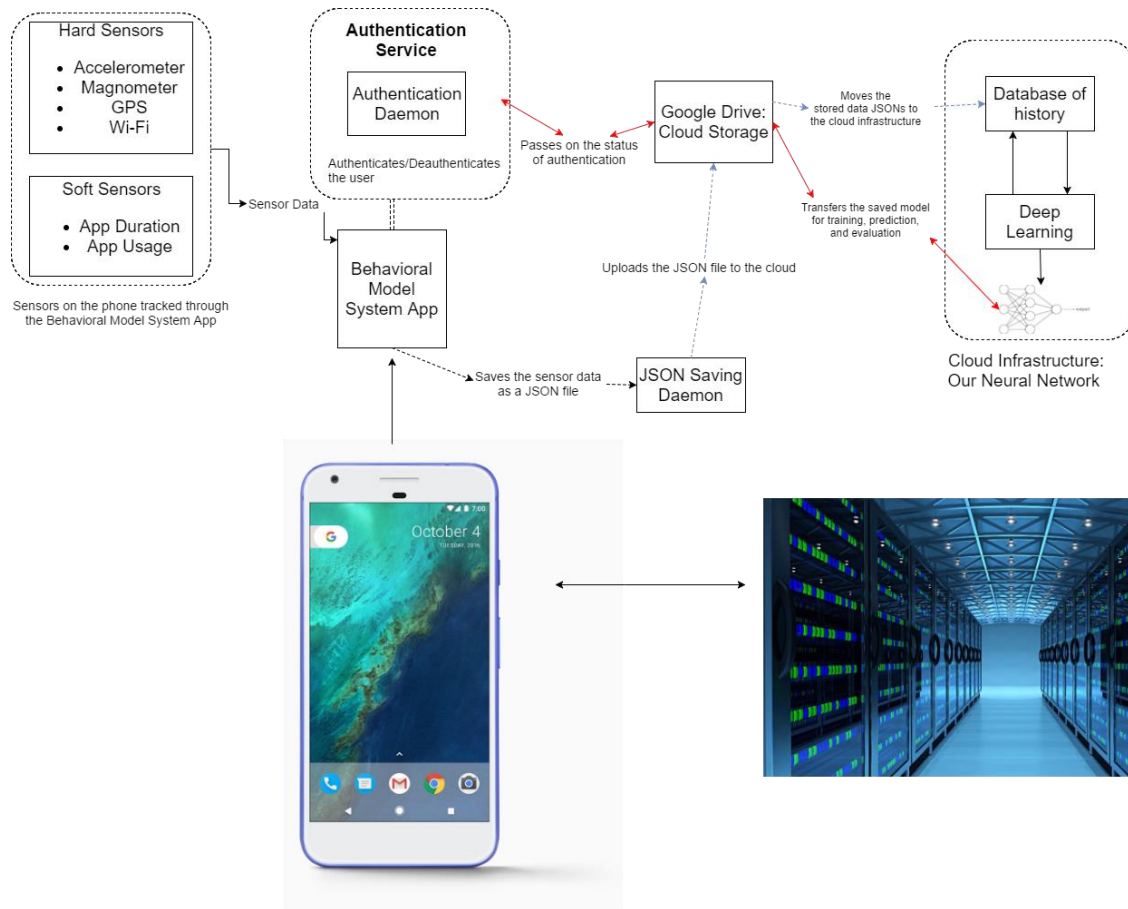


Figure 6. BMS design.

5.1 Implementing Our Authentication Modality

The implementation of our metrics used several Android APIs specific to the type and amount of data needed for each metric. As we wanted to focus on our application being optimized for more modern devices, striking a balance of widespread availability, usage, and recency, we developed on Android 6.0, Marshmallow, API Level 23. All of our metrics used APIs with practical approaches on this version of the platform, as seen in Table 8.

Metric	Android API	Description
App Usage Events	App Usage Statistics API	Retrieves statistics about the apps used on the device

Location	Awareness API	Efficiently tracks multiple sensors as a way of getting the user's context
Tilt	SensorManager Position Sensors	Access the sensors on a device such as the accelerometer
Wi-Fi	Wi-FiManager	Shows the Wi-Fi configuration of the device as well as the nearby wireless access points

Table 8. The sensors we are tracking and the API's used to do it.

5.2 Implementing Data Collection as a Background Process

These APIs shown in table 8 were used for collecting data in the background and storing it for use in our neural network. To achieve our low power goal, we collected data using Android's AlarmManager, a class in Android used for scheduling an application to be run, and used a remote server to train and create our Neural Networks.

5.2.1 Background Services

Our application is made up of several background services, each containing the API calls for fetching the location, app usage events, tilt, and touch of the device. We used Android's AlarmManager, to continuously collect data from the user on a short-interval basis. For the purpose of data collection in the background, using the AlarmManager fit our use case the best because of its ability to run in intervals even without the app open. It is also much less resource intensive compared to the alternative of a continuously running background service.

Android's AlarmManager class is used in our program through a repeating alarm for each metric that will run the data collection when the alarm goes off after a certain interval of time. In our implementation alarms were set to fire every five minutes. Unfortunately, Android 6.0 only allows "inexact" alarms, which means that Android will attempt to run multiple alarms at

the same time, potentially causing an alarm to run up to an entire cycle (the time it takes for the alarm to go off) late. This has not impacted our data collection; however it needs to be mentioned because data collection periods are roughly five minutes apart instead of exactly five minutes.

5.2.2 Data Storage

Metric	Data Being Collected	Sample Values
App Usage Events	Name of app Timestamp	Com.android.chrome 22-04-2017 06:21:15
Tilt	X axis Y axis Z axis Timestamp	5 -5 0 22-04-2017 06:21:15
Location	Latitude Longitude Timestamp	-8.93530 -50.44254 22-04-2017 06:21:15
Wi-Fi	SSID (Name of Network) Mac address (ID of Network) Signal strength Timestamp	WPI Wireless B9-1A-39-59-A5-9A -39 22-04-2017 06:21:15

Table 9. Summary of data gathered from our selected behavioral metrics.

We stored the results, time-stamped, locally on the device, as a form of a timeline of events. The data was stored in a JSON format as it was easily created locally and parsed by the server.

5.2.3 Collecting Data

The JSON files were uploaded from the phone to the user's Google Drive account. The data folder was shared to a BMS Google Drive account. To prevent synchronization problems and communications problems that may occur between our mobile clients and the server, we used

Google Drive as an intermediary to store client data and server authentication / unauthentication messages. This eliminated many problems that would occur and not undermine the underlying implicit behavioral authentication concept of our project. Google provides both android and python APIs for the Drive Service so this made this sort of communication relatively easy to design.

The Google Drive API runs as a background service on the phone. There were two mobile portions to the Google Drive API and two server portions to Google Drive API, both sets have an information sending portion and an information retrieving portion.

In the mobile application, the sending portion recorded the mobile phone's IMEI number and used it to create a folder on Google Drive. The phone then periodically uploaded collected information to this folder in the form of JSON files. This information was parsed by the server and fed into corresponding neural networks. We believe that the phone's IMEI number can be used to distinguish different users as we assume that there will only be one user per mobile phone. The mobile application also initially creates an authentication file in which the server periodically updates based on whether a user is authenticated or not. The authentication service on the phone periodically checks the file for the authentication status of the device and notifies the user whether they have been authenticated or unauthenticated. In our implementation, we found that we had to query Google Drive twice in order to properly read updates from the authentication file. We do not believe this is an issue that needs to be addressed at this stage of implementation.

5.2.4 TensorFlow Server

Our server ran Python 2.7 code using the TensorFlow 1.0 APIs. We used a fully connected, feed forward neural network with 2 hidden layers, with 8 nodes each for the sake of simplicity. We used the sigmoid activation function, which outputs values which range between 0 and 1. If the network produced values close to 1 it meant that the input values in the testing set match closely to the model.

There are a few drawbacks with this implementation. TensorFlow 1.0 APIs abstract away many functionalities that needed to be manually tuned in previous versions. One of these abstractions caused our network to save “event files” automatically in the model folder after each training session of the model. A new event file is saved after every training session. Furthermore, the information in one events file is created by appending a small amount of data to the previous events file and saving it again. When running a numerous amount of training sessions, the files started to grow to incredibly large sizes making our server run out of disk space. The space problem was patched by deleting the events files every 10 training sessions.

5.2.5 Data Representation

The input of the neural network consists of one node for each number feature, such as tiltX,tiltY,tiltZ, latitude, longitude, and others. Conversely, the strings were represented as several nodes rather than a single node in order to make it easier for the neural network to understand the input, such as individual app names and Wi-Fi BSSIDs. Since we did not want two strings with similar structure and different semantic meaning to be represented as two very close numbers, the conversion from one string to n nodes was completed by converting the string to a large number and then taking the last n bits of the number. Shown below in Table 10 are a list of

all metrics implemented and the features we extracted from them. The general features of hour and minute of the time are extracted from every metric.

Metric	Extracted Features
Tilt Specific	TiltX (float) TiltY (float) TiltZ (float)
Location Specific	Longitude (float) Latitude (float)
Wi-Fi Specific	Wi-Fi name (string) Wi-Fi bssid (string) Wi-Fi signal strength (float)
App Usage Events Specific	App name (string) Foreground/background status (int)
All	Hour (int) Minute (int)

Table 10. Metrics implemented by BMS and Features extracted from them

On initialization, the network is registered with a hashmap of feature-value pairs, which contains values for all the features that will be extracted. These values are all initialized to zero. The network takes one JSON file. In the file, each sensor object is processed and alters certain values in the hashmap. For example, if the first object is a tilt object, the tiltX, tiltY, tiltZ, hour, and minute values in the feature list are altered and the other values stay the same (as zeroes). This process assumes that when a sensor object is read in, the other sensor objects have not changed. Then, each value in the hashmap is fed into the neural network. These steps are repeated: read sensor object, update feature list, feed to network, all until the JSON file is fully input.

For the neural network's desired output, we used a 1 to indicate true data and a 0 to indicate false data. For example, subject A's model was trained positively with subject A's data using a desired output of 1, and it was trained negatively with subject B, C, and D's data using a

desired output of 0. As a result, it learned to output a trust score (float value) which is close to 1 if the given data is more similar to subject A's data and close to 0 if the given data is more similar to subject B, C, or D's data.

After the training session is finished, the model is saved automatically in TensorFlow 1.0's `regressor.fit` function. It is saved in a folder during training. After the training is done, the folder is compressed and unused until needed.

Through the implementation, the JSON files are read from the user's Google Drive, corrupted ones are removed, the model is loaded and finally trained and used for prediction based on the continuous JSON data. This process will eventually be refined, potentially in future work, but we intend for the process to be completely automated through a more finalized release of BMS.

6 Results

We decided that we would need to test both our systems effectiveness in authentication and the systems' impact on phone battery life. We used the JSON files uploaded by the test phones and separated into a training and testing set in order to evaluate the neural networks, and we used an app called GSAM Battery Monitor[44] to collect metrics about the battery life of test phones running the BMS application and not running the BMS application in both sitting and moving scenarios.

6.1 Discussion of Behavior of Users Surveyed

To understand some of the results, we first needed to explain some of the nuances in our collection of data. We used 4 test subjects for data collection, subject A, subject B, subject C, and subject D. All subjects were undergraduate senior computer science students attending Worcester Polytechnic Institute.

- Subjects A, B and C live in the same apartment and go out for dinner together almost daily. Subjects A, B and C live in the same apartment and go out for dinner together almost daily.
- Subjects B and C have adjacent rooms and their workspaces are within a few feet of one another.
- Subject D often visits the apartment of subject A, B and C and often is in the rooms of subjects A and C.
- Subjects A and B share a 1-hour class together in which they also meet for group projects
- Subjects C and D share a 2-hour class together

- Although this should not be a problem, subjects C and D share the same phone model which may play a factor that we are not aware of.

6.2 Battery Tests

The GSam Battery Monitor app was used to monitor battery drain on an Android phone with and without the BMS app running. This collected the percentage of the battery used by each app and the rate of change of the battery. A Nexus 6 smartphone, running Android 7.1.1 Nougat, was left in the same location for 4 hours. The only apps installed on the Nexus 6 other than the stock apps were the GSam Battery Monitor and the BMS app. App usage events, Wi-Fi, location, and tilt data were pulled at intervals of 60 seconds, 60 seconds, 60 seconds, and 30 seconds, respectively. The difference in battery drain are shown below in Table 11. With BMS running, the smartphone only used an additional 2% of battery life in 4 hours.

Further testing is required to determine if the results are accurate to real-world scenarios. Since the phone remained stationary and unused during the testing period, the phone would have recorded less data in terms of app usage. As a result, the stored JSON files would have been smaller and required less battery to write and upload. In addition, some smartphone sensors may go into a “sleep” mode when the phone is not active. If this were the case, BMS would drain battery faster than expected from the battery test.

	With BMS	Without BMS
Battery Drain	10%	8%

Table 11. Battery drain with and without the BMS app.

6.3 Authentication Results

Data for four subjects was collected over a period of approximately 11 days. For each subject, his neural network model was tested on day x after being positively trained to output 1

on his data and negatively trained to output 0 on all other subjects' data on all days before day x . For example, a model for subject A's 5th day could be trained on subject A's data positively and subject B, C, and D's data negatively for all data up to that day, days 1 through 4. Predictions are then made for the 5th day. For each subject, subject A's model will take that subject's data and output a predicted trust score. This score should be close to 1 for subject A's data and close to 0 for subject B, C, and D's data. Subsequently, the 5th day, if trusted, is added to the model as additional training. At this point, the neural network model has incorporated training from days 1 through 5. The process is repeated to make predictions for the 6th day, and so forth. This reflects real-world conditions, where the network continually receives new data to train on and must give a predicted authentication score for the current day.

Features from all metrics: Wi-Fi, location, tilt, and app usage events, were used in training. For the specific features extracted from each metric, please see Table 10. Below (Figures 7 – 10) are graphs of subject A, B, C, and D's models, respectively. The predictions for each subject on each day are graphed. Each line represents a single subject's predicted trust score and shows how it changed after each day.

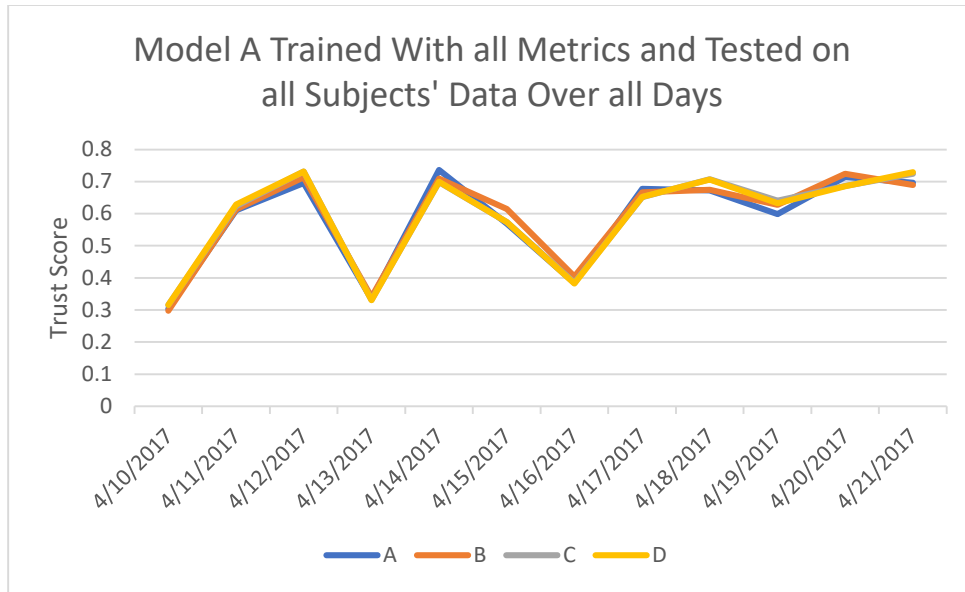


Figure 7. Subject A's neural network, trained to output 1 when given subject A's data and 0 when given subject B, C, or D's data. Training data consisted of all metrics: tilt, Wi-Fi, location, and app usage data. The predicted trust score is plotted for each subject on each day.

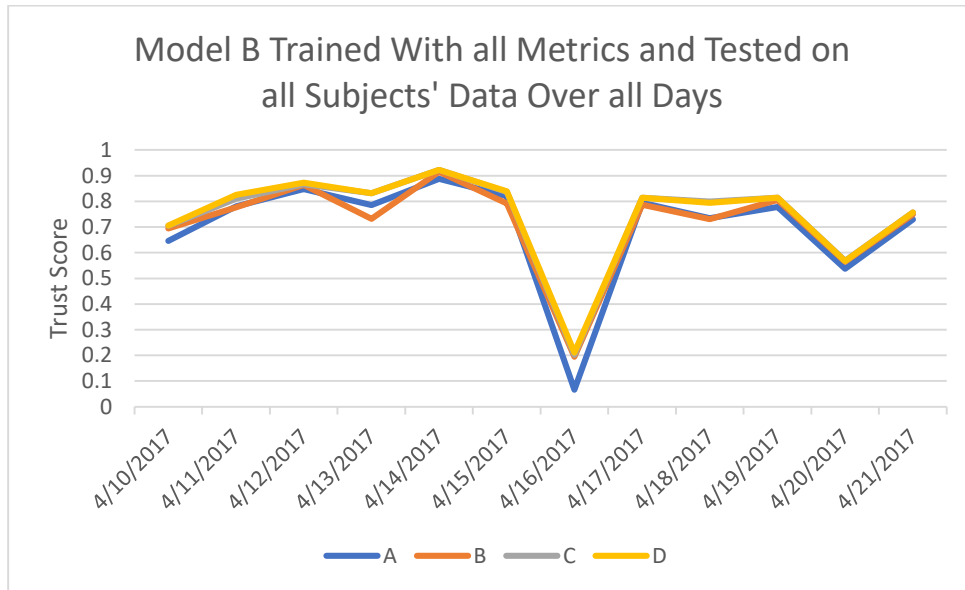


Figure 8. Subject B's neural network, trained to output 1 when given subject B's data and 0 when given subject A, C, or D's data. Training data consisted of all metrics: tilt, Wi-Fi, location, and app usage data. The predicted trust score for all subject's data on each day was plotted.

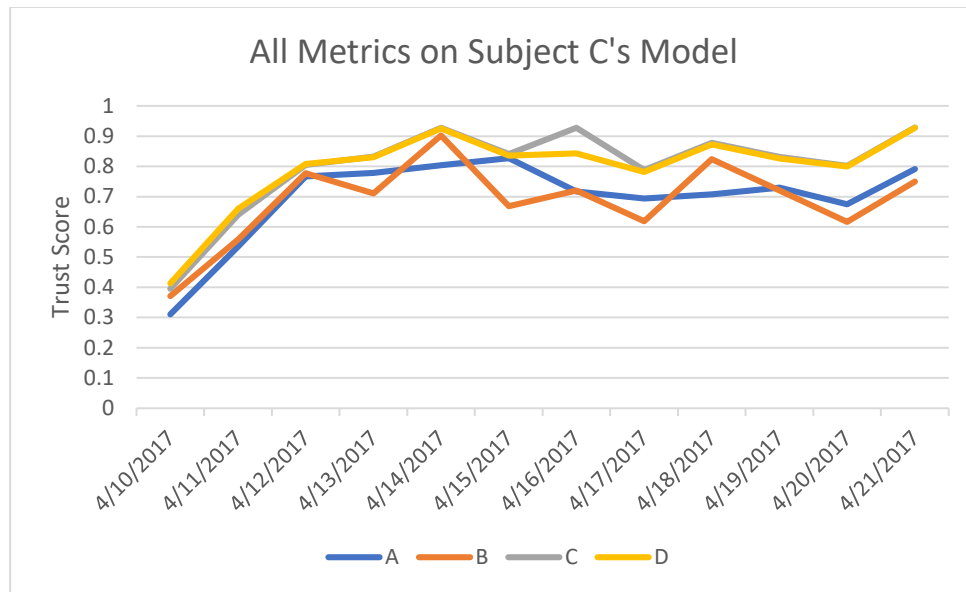


Figure 9. Subject C's neural network, trained to output 1 when given subject C's data and 0 when given subject A, B, or D's data. Training data consisted of all metrics: tilt, Wi-Fi, location, and app usage data. The predicted trust score for all subject's data on each day was plotted.

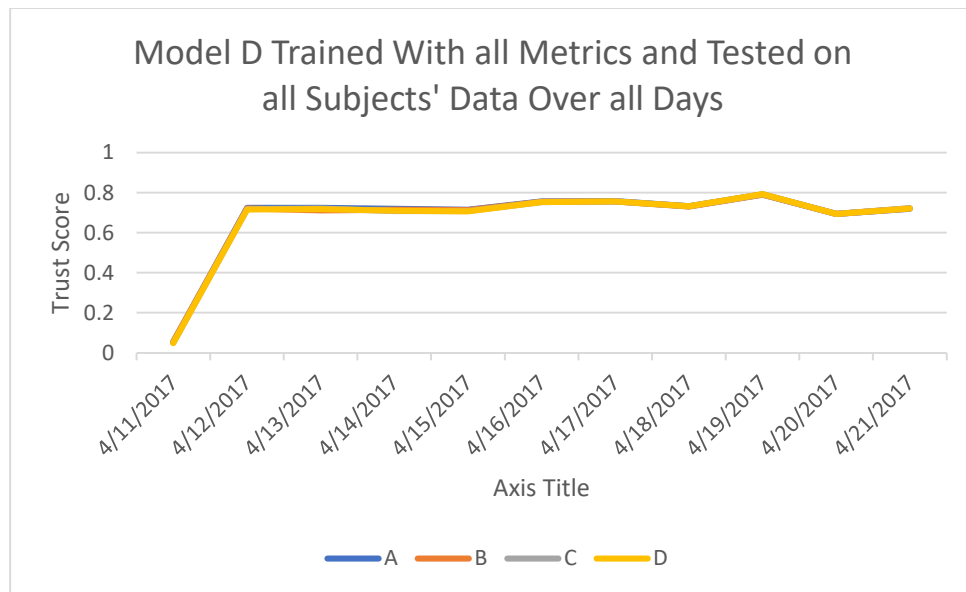


Figure 10. Subject D's neural network, trained to output 1 when given subject D's data and 0 when given subject A, B, or C's data. Training data consisted of all metrics: tilt, Wi-Fi, location, and app usage data. The predicted trust score for all subject's data on each day was plotted.

In each person's graph, they all followed generally the same trend with user behavior, within a few hundredths of fitting to one another's user behavior. Only in case C did any single user's behavior fit their own behavior better than other people fit their own behavior. Despite the users leading very different behaviors and lives, especially with app usage behavior. Below,

Table 12. Day 11 Confusion Matrix shows a better breakdown of the authentication values on the last day (day 11) of testing.

		Model			
Testing		A	B	C	D
	A	0.696176851	0.72920086	0.790976558	0.718816646
	B	0.689681277	0.749293778	0.749973414	0.72049831
	C	0.724192657	0.756450769	0.92878729	0.722945101
	D	0.729186394	0.756887059	0.927878674	0.721123157

Table 12. Day 11 Confusion Matrix of training on all metrics.

After training on days 1 through 10, the network gave mixed results when tested on day 11. Of the four models, model C (Table 12, 3rd column) performed the best, giving its subject (subject C) the highest trust score, 0.9287. It is important to note that this score exceeded the second highest score, 0.9278, by a difference of only 0.0009. With such a small margin, model C was not able to differentiate data from subjects C and D.

Models A, B and D were unable to assign their subject the highest trust score. Model A (Table 12, 1st column) gave subject A the third highest trust score of 0.6961, noticeably lower than 0.7241 and 0.7291, the scores given to subjects C and D, respectively. Model B (Table 12, 2st column) gave subject B the third highest trust score of 0.7492, slightly lower than 0.7564 and 0.7568, the scores given to subjects C and D respectively. Finally, model D (Table 12, 4th column) gave subject D the second highest trust score, only slight lower than 0.7229, the score given to subject C. It was disappointing to find that using all metrics and 10 days of training data, models A, B, and D did not each correctly score their own subject as the most likely to be authenticated.

6.3.1 Accuracy

In subject A's model (Figure 7), the network produced very similar trust scores for all subjects at nearly every day, but there was more deviation in the trust scores near the end of the testing period. While we expected the network to differentiate more as more training data was made available, subject A's model was still unable to give subject A's testing data the highest trust score near the end of the training period. The results for subject B (Figure 8) and subject D's (Figure 10) models are similar, with the network unable to give its user the highest trust score and give other subjects a lower trust score.

Subject C's neural network model (Figure 9) was the most successful in differentiating between the subjects and predicting the highest trust score for subject C. The scores across

different subjects generally deviated more as more training was available, showing that the network was able to differentiate the behavior between the four subjects. However, the model gave subject D almost exactly the same trust score on several days. Depending on the authentication score threshold we set, the model could have authenticated subject D using subject C's smartphone.

6.3.2 Patterns

For all models (Figures 7 - 10), the trust scores on subjects C and D are very similar. Furthermore, the score of model C on D is very close to the score of model C on C, and the score of model D on C is very close to the score of model D on D. We can infer that subjects C and D are very similar, as measured with the network's trust score. This was expected, since subjects C and D share a two hour class twice a week, and subject C occasionally works in subject D's room. The location and Wi-Fi networks would have been similar for both subjects during these times.

In addition, the trust scores on subjects A and B are similar for all models. This was expected because they share a one-hour class and meet for project work for several hours at a time. Also, they use similar apps, including Messenger and Telegram. Therefore, in many ways it is consistent that their evaluations across each other's models are similar.

6.3.3 Comparison with Users' App Usage Statistics

The below table shows the most commonly used applications used by each subject. There are a few commonalities between each subject and their most used applications, but there are also many differences. We can qualitatively see that there are differences between each user, but they are not reflected in the models shown in the next sections.

Subject	5 Most Frequently Used Applications
Subject A	<ol style="list-style-type: none"> 1. Manga Mobile 2. Google Chrome 3. Facebook 4. Telegram 5. YouTube
Subject B	<ol style="list-style-type: none"> 1. Reddit 2. Google Chrome 3. Facebook 4. Telegram 5. Music
Subject C	<ol style="list-style-type: none"> 1. Textra 2. Google Chrome 3. Tinder 4. Reddit 5. Telegram
Subject D	<ol style="list-style-type: none"> 1. Duel Links 2. Google Chrome 3. Quora 4. Photos 5. Facebook

Table 13. List of most commonly used applications by Subject

6.4 Individual Metrics

Data for the four subjects was compiled in the same way in consecutive designs of the model. Tracking individual metrics at a time, we systematically tested each metric on each user. For the specific features extracted from each metric, please see Table 10. Further, a subject's model was negatively trained on behavior against the other subjects' three models, training those behaviors to not be considered valid since it is not from that user. The charts below show the results of testing all subjects' models, using the individual metrics of app usage (Figures 11-14), location (Figures 15-18) and Wi-Fi (Figures 19-22), and using training data from all subjects.

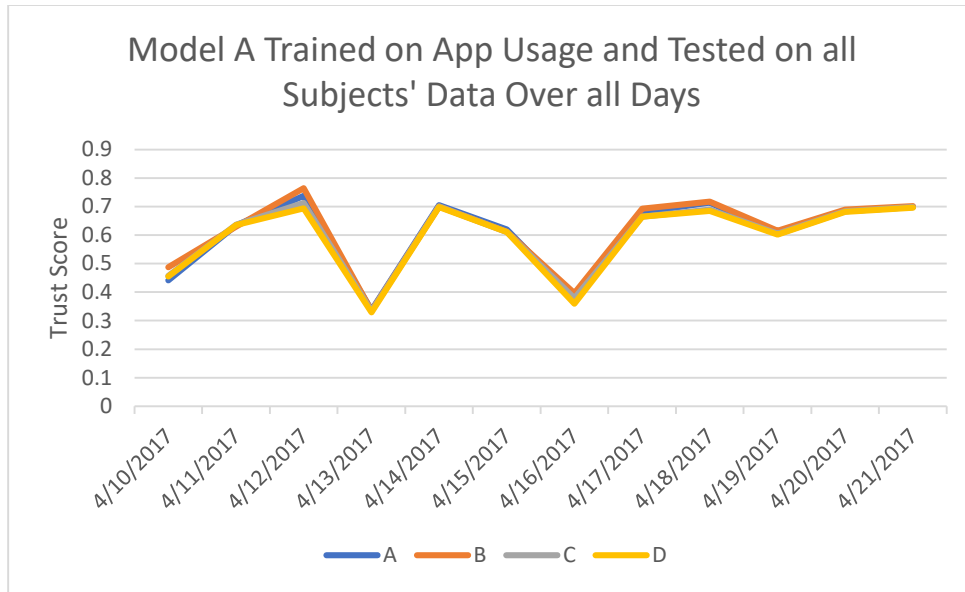


Figure 11. Subject A's neural network, trained to output 1 when given subject A's data and 0 when given subject B, C, or D's data. Training data consisted of only app usage data. The predicted trust score for all subject's data on each day was plotted.

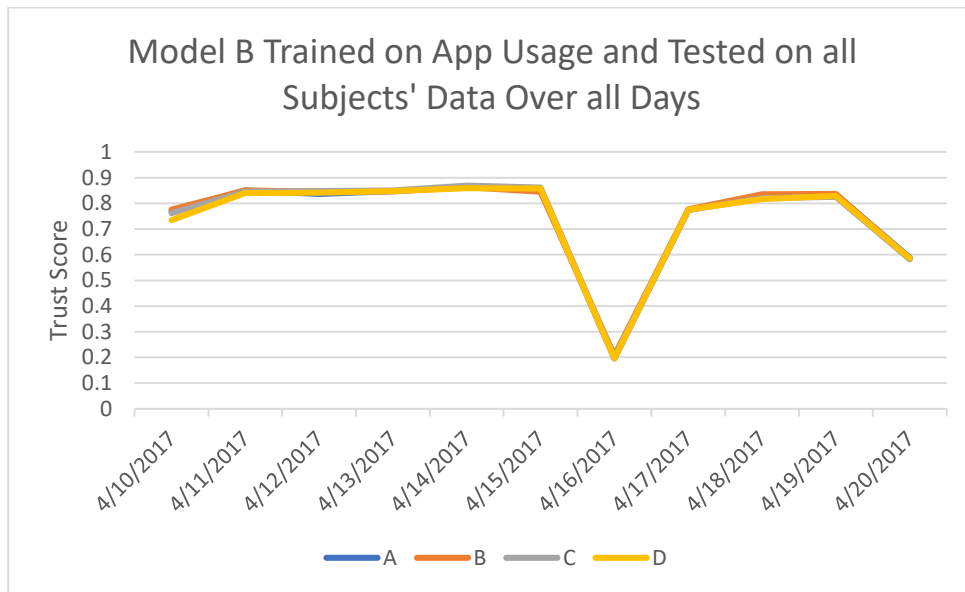


Figure 12. Subject B's neural network, trained to output 1 when given subject B's data and 0 when given subject A, C, or D's data. Training data consisted of only app usage data. The predicted trust score for all subject's data on each day was plotted.

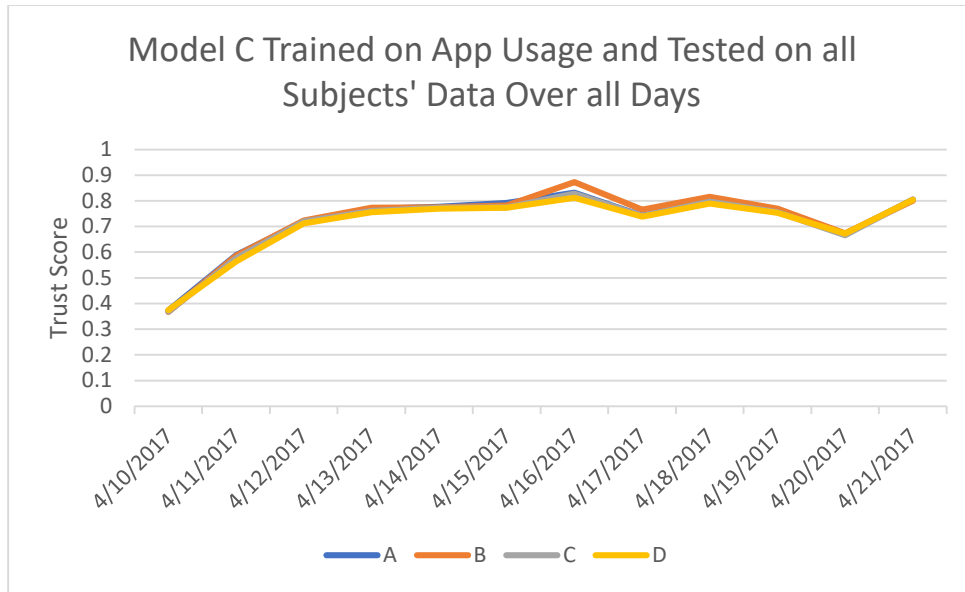


Figure 13. Subject C's neural network, trained to output 1 when given subject C's data and 0 when given subject A, B, or D's data. Training data consisted of only app usage data. The predicted trust score for all subject's data on each day was plotted.

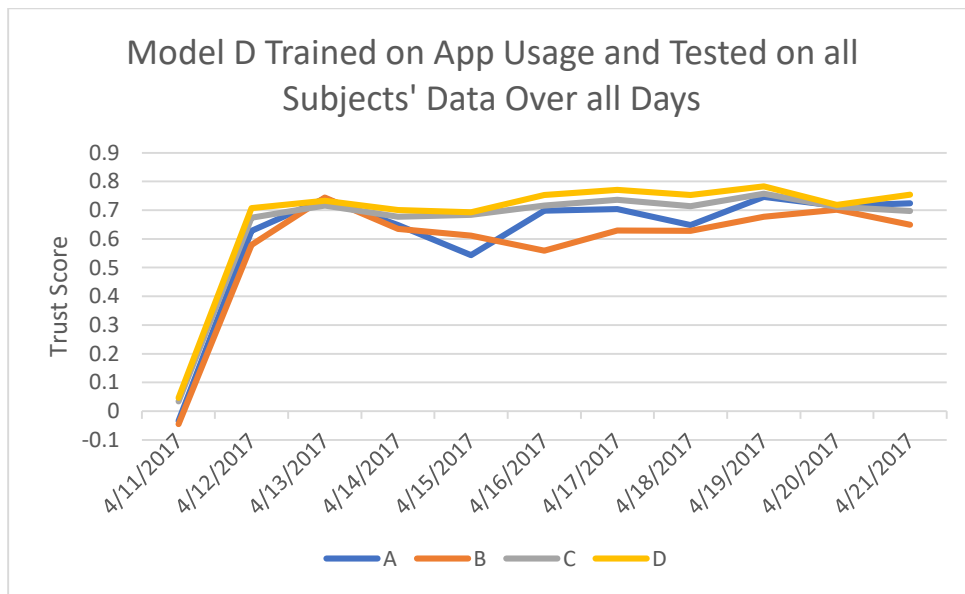


Figure 14. Subject D's neural network, trained to output 1 when given subject D's data and 0 when given subject A, B, or C's data. Training data consisted of only app usage data. The predicted trust score for all subject's data on each day was plotted.

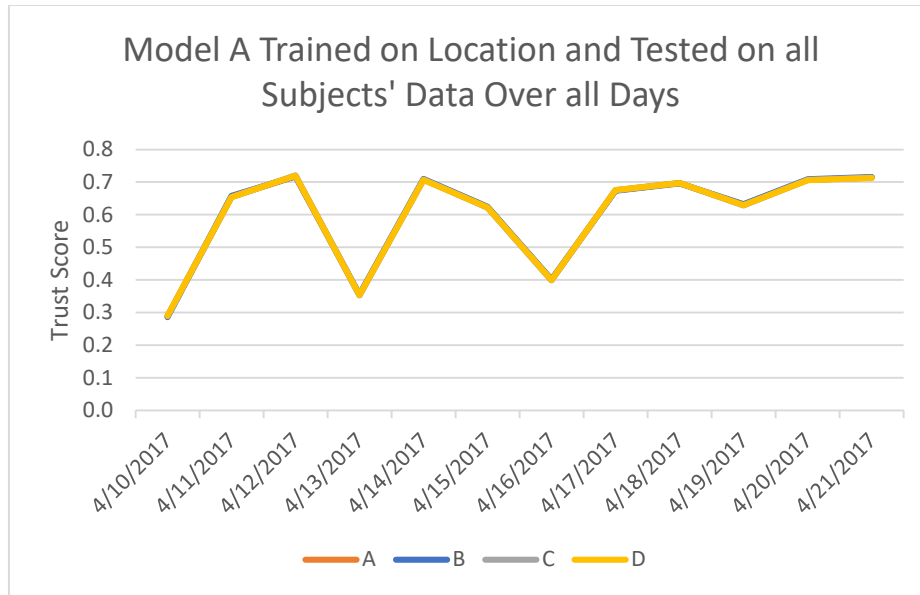


Figure 15. Subject A's neural network, trained to output 1 when given subject A's data and 0 when given subject B, C, or D's data. Training data consisted of only location data. The predicted trust score for all subject's data on each day was plotted.

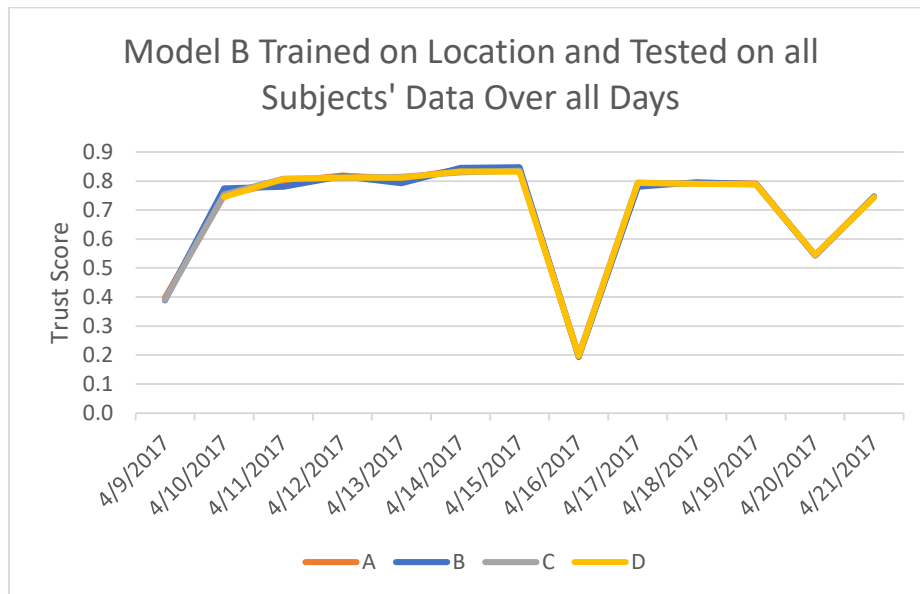


Figure 16. Subject B's neural network, trained to output 1 when given subject B's data and 0 when given subject A, C, or D's data. Training data consisted of only location data. The predicted trust score for all subject's data on each day was plotted.

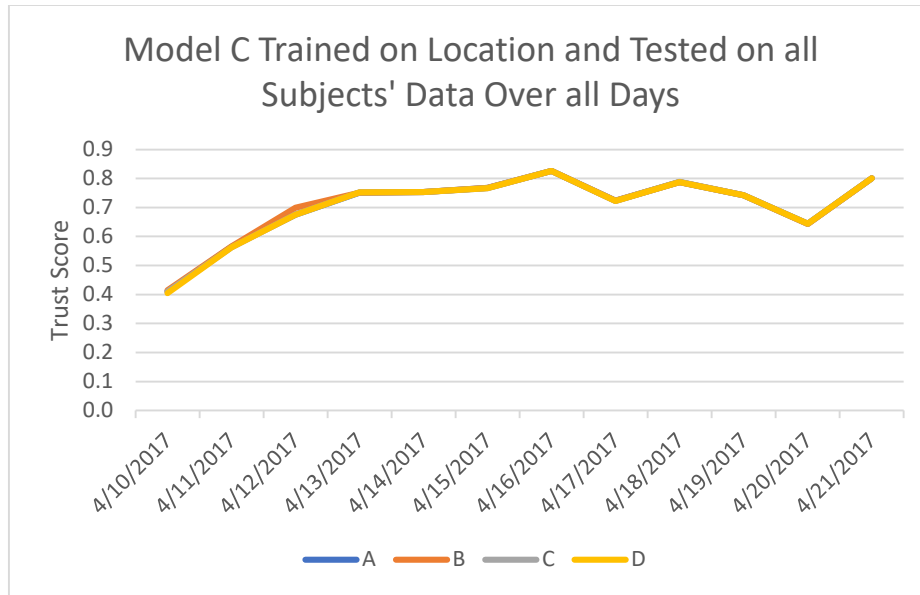


Figure 17. Subject C's neural network, trained to output 1 when given subject C's data and 0 when given subject A, B, or D's data. Training data consisted of only location data. The predicted trust score for all subject's data on each day was plotted.

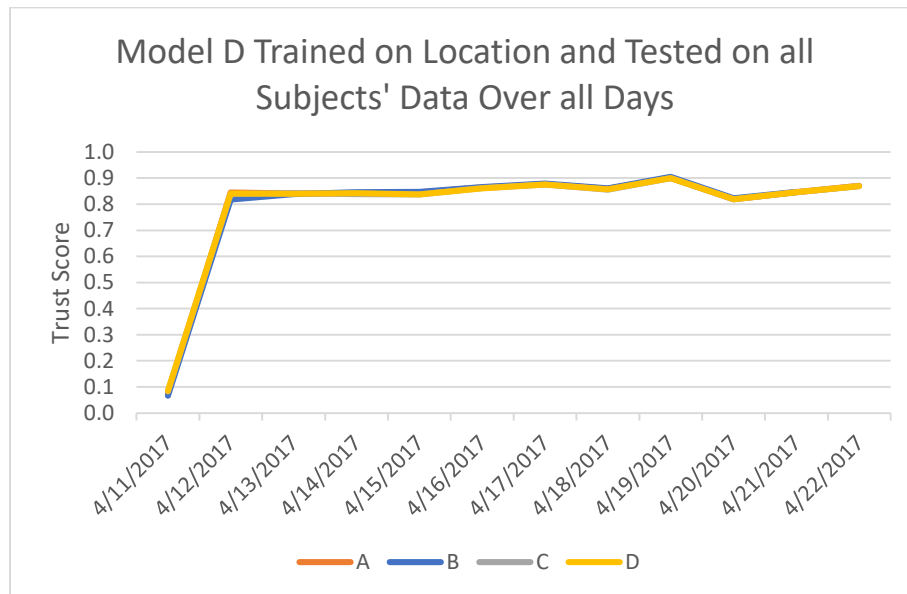


Figure 18. Subject D's neural network, trained to output 1 when given subject D's data and 0 when given subject A, B, C's data. Training data consisted of only location data. The predicted trust score for all subject's data on each day was plotted.

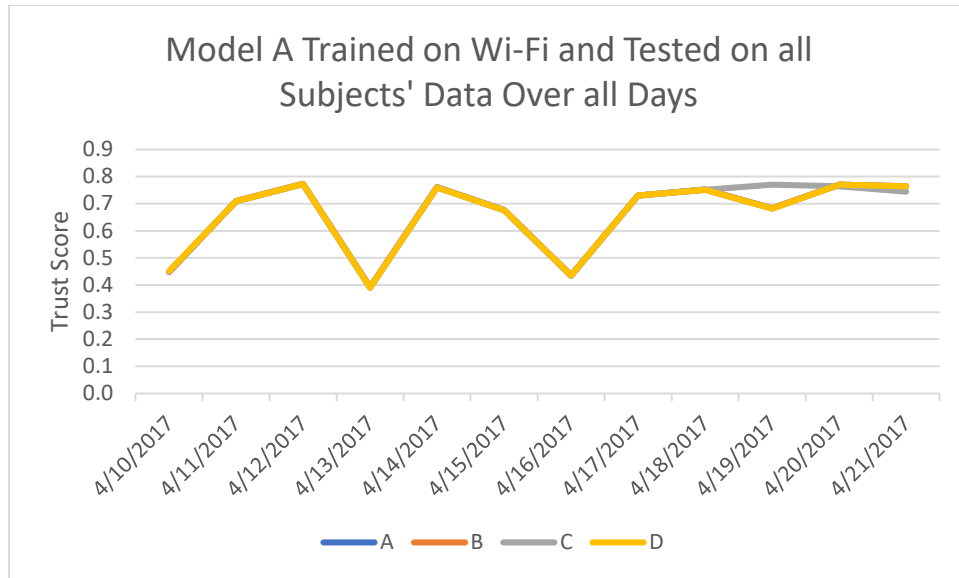


Figure 19. Subject A's neural network, trained to output 1 when given subject A's data and 0 when given subject B, C, or D's data. Training data consisted of only Wi-Fi data. The predicted trust score for all subject's data on each day was plotted.

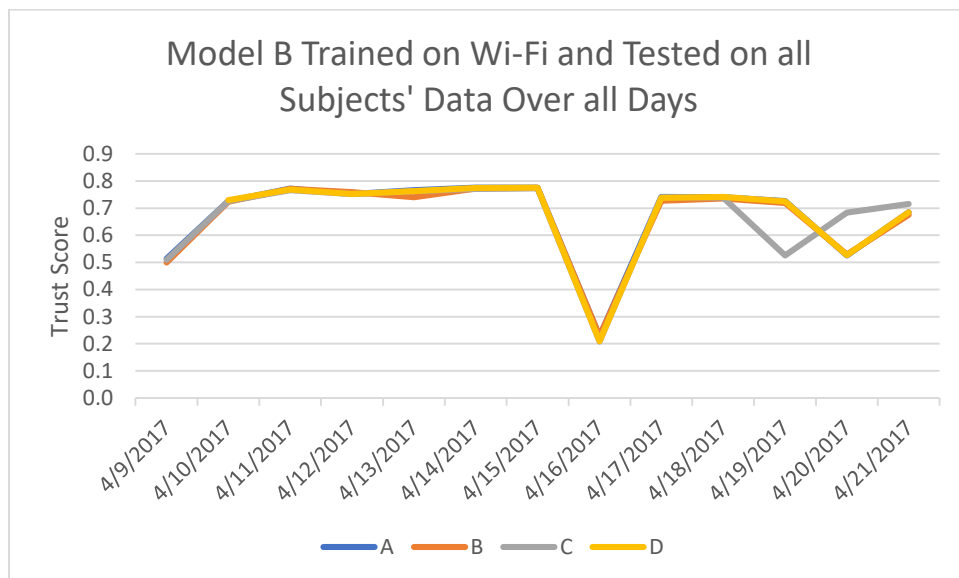


Figure 20. Subject B's neural network, trained to output 1 when given subject B's data and 0 when given subject A, C, or D's data. Training data consisted of only Wi-Fi data. The predicted trust score for all subject's data on each day was plotted.

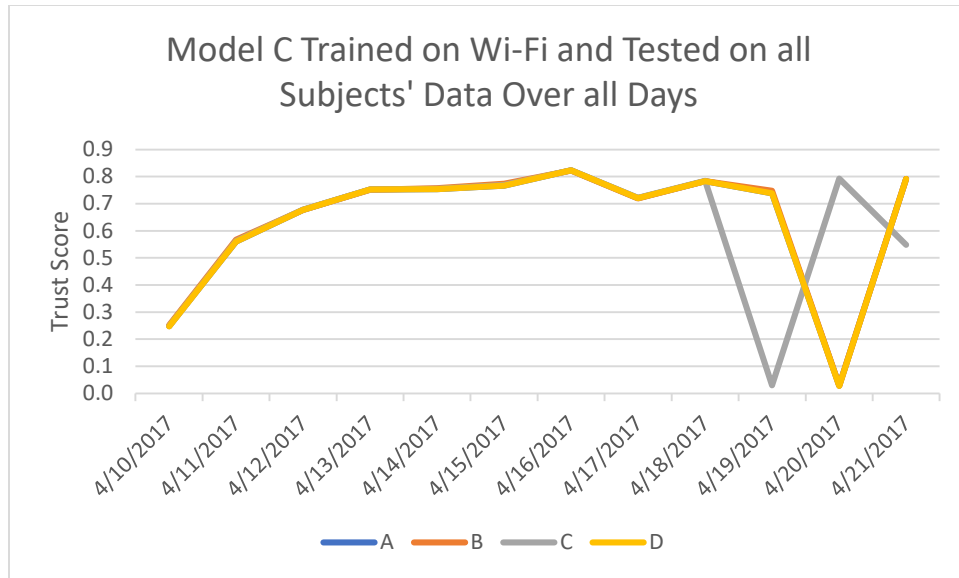


Figure 21. Subject C's neural network, trained to output 1 when given subject C's data and 0 when given subject A, B, or D's data. Training data consisted of only Wi-Fi data. The predicted trust score for all subject's data on each day was plotted.

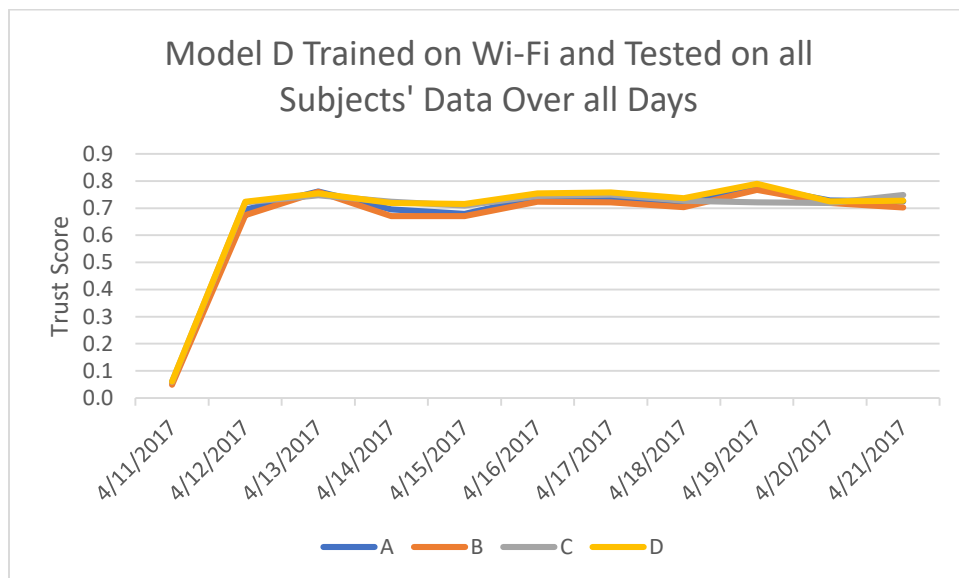


Figure 22. Subject D's neural network, trained to output 1 when given subject D's data and 0 when given subject A, B, or C's data. Training data consisted of only Wi-Fi data. The predicted trust score for all subject's data on each day was plotted.

6.4.1 Accuracy

As consistent with the previous results, Subject A and B remain entirely unpredictable, even with negative data fed previously to his model to discourage authenticating other users. Likely what happens for this user is that his behavior is erratic enough to not have identifying features, and during normal days of theirs, their behavior does not fit their respective models

strongly. Subjects C and D's results are consistently very much like one another, and they evaluate similarly on each other's models as well, with similarities almost within a margin of error in similar behavior. Ideally, these would be more disparate, but their behavior are indeed incredibly comparable.

6.4.2 Comparison of Individual Metrics against All Metrics

Subjects A and B had similar models in all four of the model types: all metrics, only location, only Wi-Fi, and only app usage. In this case, this would imply that location, Wi-Fi, and app usage were all in agreement, and all metrics are equally inaccurate in identifying these two test subjects.

Subject C's model trained using all metrics performed better and differentiated users better than models trained with only one metric. In this case, the use of all metrics together was more effective than any individual metric. Since a trust score can also be obtained by summing scores from multiple independent sources, it was encouraging to find that the neural network was able to find correlations between different metrics.

For subject D, the model trained using only app usage performed better and differentiated users better than models trained with only Wi-Fi, only location, and all metrics. As a result, app usage is a strong predictive factor of correct authentication for subject A. It is unusual that the network performed better with less data available. It is possible that the other less predictive metrics, Wi-Fi and location, caused the model with all metrics to "average out" to a worse accuracy.

6.5 Discussion of Results

Our results overall, compared to what we were hoping for and expecting, have been underwhelming. There are many reasons we rationalize these results, and see them as helpful to future implementations based off this design and implementation.

6.5.1 Difficulties with Artificial Intelligence

Our resources in implementing our AI were heavily limited, as our efforts were split among app development, automation, research, and AI design. It is very possible that with our efforts so divided, we ended up overlooking an issue with inputting data to the network, or network design in general that would otherwise keep us from getting fitting results for all of our behaviors of the day.

6.5.2 Similarities with Behavior

As discussed earlier in the chapter, many of the test subjects have incredibly similar behaviors and routines. All of them are in the same grade, work in the same places, the same classes and group projects, and attend the same college. Even some activities outside of school, such as meals, trips, and other minor behaviors were shared among us all, so to suggest that the network trusted all our behavior too much and tried to both positively and negatively fit to

6.5.3 Improving our Network

A day-of-week feature should be added to the neural network, to improve the notion of weekly routines. For example, a college student may have classes scheduled every Monday at noon or a person might have leave work early every Friday. If only the hour and minute values are used in the neural network, it would be trained to sometimes associate the classroom's

location with 12:00 pm on the days the student had class, contradicting the association on days without class.

Additionally, a recurrent neural network (RNN) should be tested. In a RNN, the output is affected by previous inputs as well as the current input, dealing with sequential data more effectively. For instance, a daily routine could be leaving home, getting breakfast, and then going to work. The sequence of events is more meaningful than the exact times during which these events occurred.

It is possible that the data given to the neural network is too specific. Instead of the x,y, and z orientations of the phone, labels such as "upright" or "lying flat" might be more effective. In place of an app's name, we could use an app category, such as "game", "music", or "photography". Similarly, a location could be represented as categories such as "restaurant", "store", or "gas station". If a person went to a different restaurant every night, the network would differentiate between the longitude and latitude values but ignore the motivation behind visiting the location.

6.5.4 Final Words

Although our results are underwhelming we believe that a system like this could work. We cannot make a strong assertion to the viability of our system. We would need a more diverse set of test subjects and a more developed artificial intelligence to make any stronger claims.

7 Future Works

To fully understand the project we have implemented, we offer our vision for the future of the project, and rationalize why such efforts are viable and the appropriate directions for this project to take.

7.1 Modular Implementation Allowing User to Select Various Combinations of Metrics

More than anything, the user must be able to trust the platform that they use to authenticate themselves to their electronic devices. Without designing each aspect of BMS to be toggleable, therefore disallowing the user complete control of what is being used to authenticate them, the user is likely to not feel comfortable giving the entirety of the model complete permissions; in short, the user needs to be able to choose the metrics they are comfortable in sharing with BMS. The only way this succeeds is in a modular design of the implementation of BMS. Naturally, going forward, we find the model to be usable in increasingly complex settings, with potentially more intrusive metrics, on both phone usage and battery life, so to add these metric modules, we would want the user to be able to shut them off as they deem necessary to get the model that they want.

7.2 Negative Sample Models

As this is a security application, implicitly authenticating the user, it is essential that we are accurate with authenticating our user on their phone and no other. In future implementations, we would gather the user data that already exists, and with the user's consent we would anonymize that data and add it to a repository of non-matches, based on a hashed user ID. That gathered data would later be introduced to other users' models, to see if any of the other

users' models would output an "authenticated" or significantly trusted score from the data that is not meant to authenticate those models. If this happens, the user will be notified that their model is not yet strong enough, and they have to keep building it in order for it to improve further. If this persists across several weeks, the app and server would later inform the user that their interactions with the phone are particularly hard to distinguish from others, and that it is recommended they add more metrics if available or use a secondary authentication method like a trusted device or explicit authentication method. This way, we can ensure the user is informed as to whether or not their model is particularly secure or unique, and that they should feel decently trusting of the technology.

7.3 Multi-User Support on Shared Devices

In rationalizing the idea of the neural network being able to distinguish between users and provide advice about using more metrics to uniquely identify the user, a more profound idea emerged. If our model is able to distinguish between several other users' data at once, it would be similarly feasible to set up two or more instances of BMS on a phone, one for each respective user. Every user would have a training phase, and at every point along the model creation phase, preferably through a fingerprint, data would be saved to the respective user. Once all instances of BMS are in place, a user could be recognized merely by interactions with the phone that it finds to be matching a particular user, locking out certain interactions, accounts and data tethered to other users on the phone.

7.4 Distinguishing Group of Friends with Many Shared Behavioral Patterns.

This level of user granularity does not have to be tethered to just one device. In the exemplary case of two friends regularly going out to eat together, they could very easily link their

profiles in a future iteration of BMS to relate location data and potentially each other's behavioral models for a time, so both of their phones will recognize when the friend is using it, or the shared behavior between both can be matched as an identifying metric for the model; the fact that you and your friend's phones are in the same location for a time can be taken as positive input for the authentication metric. When friends are in a location where a user are expected to be according to a model, such metrics can also immediately de-authenticate the user's phone.

7.5 Multi-Device Connection and Internet of Things

Just as a phone and behavioral model can't be expected to just handle one set of behaviors at any point, a behavioral model should expect the user to not contain all of their digital interactions to one device. Tie-ins to read from connected users accounts that could provide more data would be invaluable merely from the number of and time of interactions with that account. This technology has many use cases, outlined in examples below.

7.5.1 Integrating Metrics Gathered from Home Assistants

Home Assistant Devices such as Google Home and Amazon Alexa that have been taking the consumer market by storm and growing significantly in popularity over the past few years. These devices allow users to augment their daily lives by allowing them to ask questions, play music, schedule events, etc. These devices, associated with certain user accounts, offer a wide variety of data and real-world interactions a user has with the device. These interactions are saved, and note the time of day, length of interaction, and the parsed query. With all of this very contextual, very precise vocal data gathered with these assistants, the data can be uploaded to the behavioral model as a metric, and verified based on the "Trusted Voice" Google uses for authentication on the mobile device.

7.5.2 Supporting Smartwatch Interactions

A "Trusted Voice" works for many situations, but in many ways a much more personal interaction with the user is necessary. These personal monitoring devices can build off our behavioral model system through the addition of smartwatches, which can monitor a user's gate, heart rate, GPS, nearby Bluetooth and Wi-Fi devices (with specific attention to the phone's Bluetooth connection for proximity monitoring). Coupled with the data the phone can receive, a more complete model of the user's behavior becomes available.

7.5.3 Supporting Desktop Interactions

Integrations with other devices should not just end at the wrist or in the Google ecosystem. The integrations can extend to desktop interactions, and behavioral app statistics can be used to complement the data already gathered from phones to build a more complete model of user behavior.

7.5.4 Supporting Other Mobile Devices

Not every user contains their mobile interactions to one device. Using an account-based timeline to tether multiple device interactions to the same person will give a more complete user picture, and authenticate different devices at different times based on user behavior.

7.5.5 User Timeline

Ideally, these smaller concepts would be unified under a singular BMS account per individual, and would have a fairly complete timeline of all the interactions with the user. [OBJ]

7.6 Evaluating Different Neural Networks

We used a feedforward network for our implementation because it was simple to implement and worked for finding patterns in data. However, feedforward networks do not have

a strong temporal component to them. We believe that testing networks with dynamic temporal behavior such as Recurrent Neural Networks, which deal better with one-dimensional adjacency, would better help with finding behaviors that occur in a certain order at approximate times of the day, such as going to get coffee and then going to work.

8 Conclusions

Explicit authentication methods are proving to be insufficient for the personal data stored on smartphones. We propose the Behavioral Model System (BMS), a continuous authentication system that uses deep learning to authenticate in a way that is accurate and constantly adjusting to the user's routines. The BMS mobile application tracks the Wi-Fi, location, app usage, and tilt of the phone and uploads the information to Google Drive. The information is then pulled from Google Drive and fed into the TensorFlow Server to build models of each user's behavior. Both the mobile application and TensorFlow Server are modular enough to add and remove metrics in the future. The BMS mobile application does not have a significant impact on phone battery life. The results on our TensorFlow server are a little underwhelming. This could be due to several factors such as the similarities between our test subjects and our inexperience with TensorFlow. Future work includes giving the user control of collected metrics, notifying the user if their model is inaccurate, multi-user authentication, and group behavior.

Bibliography

- [1] S. P. Miller, B. C. Neuman, J. I. Schiller, and J. H. Saltzer, "Kerberos authentication and authorization system," in *In Project Athena Technical Plan*, 1987.
- [2] M. Frank, R. Biedert, E. Ma, I. Martinovic, and D. Song, "Touchalytics: On the Applicability of Touchscreen Input as a Behavioral Biometric for Continuous Authentication," *IEEE Trans. Inf. Forensics Secur.*, vol. 8, no. 1, pp. 136–148, Jan. 2013.
- [3] T. Masui, "EpisoPass: Password Management based on Episodic Memories," in *Proceedings of the 21st Workshop on Interactive Systems and Software (WISS2013)*, pp. 109–114.
- [4] H. Crawford, K. Renaud, and T. Storer, "A framework for continuous, transparent mobile device authentication," *Comput. Secur.*, vol. 39, Part B, pp. 127–136, Nov. 2013.
- [5] E. Shi, Y. Niu, M. Jakobsson, and R. Chow, "Implicit Authentication through Learning User Behavior," in *Information Security*, 2010, pp. 99–113.
- [6] H. Khan, A. Atwater, and U. Hengartner, "Itus: an implicit authentication framework for android," in *Proceedings of the 20th annual international conference on Mobile computing and networking*, Maui, Hawaii, USA, 2014, pp. 507–518.
- [7] A. Rahmati, A. Qian, and L. Zhong, "Understanding Human-battery Interaction on Mobile Phones," in *Proceedings of the 9th International Conference on Human Computer Interaction with Mobile Devices and Services*, New York, NY, USA, 2007, pp. 265–272.
- [8] C. Liu, "Worldwide Internet and Mobile Users eMarketer's Updated Estimates for 2015," Aug. 2015.
- [9] D. Mekouar, "Americans Check Their Phones 8 Billion Times a Day « All About America." .
- [10] K. P. C. B. www.kpcb.com, "2015 Internet Trends." [Online]. Available: <http://www.kpcb.com/blog/2015-internet-trends>. [Accessed: 25-Apr-2017].
- [11] A. Aviv, "Smudge Attacks on Smartphone Touch Screens."
- [12] X. Suo, Y. Zhu, and G. S. Owen, "Graphical passwords: a survey," in *21st Annual Computer Security Applications Conference (ACSAC'05)*, 2005, p. 10 pp.-472.
- [13] "Android 6.0 APIs | Android Developers." [Online]. Available: <https://developer.android.com/about/versions/marshmallow/android-6.0.html>. [Accessed: 25-Apr-2017].
- [14] "Iris Recognition | Galaxy S8 Features |," *Iris Recognition | Samsung Australia*. [Online]. Available: <http://www.samsung.com/au/iris/>. [Accessed: 25-Apr-2017].
- [15] "Set up your device for automatic unlock - Nexus Help." [Online]. Available: <https://support.google.com/nexus/answer/6093922?hl=en>. [Accessed: 25-Apr-2017].
- [16] B. Cantafio, "Security vs. Convenience," Apr. 2004.
- [17] A. Carman, "Google could replace some passwords with a 'trust score' by the end of the year," *The Verge*, 23-May-2016. [Online]. Available: <http://www.theverge.com/2016/5/23/11749938/google-android-password-trust-score-api-io>. [Accessed: 26-Apr-2017].
- [18] G. Cluely, "The top 10 passcodes you should never use on your iPhone," *Naked Security*, 14-Jun-2011. .
- [19] A. Hern, "Hacker fakes German minister's fingerprints using photos of her hands," *The Guardian*, 30-Dec-2014.
- [20] E. Nakashima, "Hacks of OPM databases compromised 22.1 million people, federal authorities say," *Washington Post*. [Online]. Available: <https://www.washingtonpost.com/news/federal-eye/wp/2015/07/09/hack-of-security-clearance-system-affected-21-5-million-people-federal-authorities-say/>. [Accessed: 18-Dec-2016].

- [21] W. Meng, D. S. Wong, S. Furnell, and J. Zhou, "Surveying the Development of Biometric User Authentication on Mobile Phones," *IEEE Commun. Surv. Tutor.*, vol. 17, no. 3, pp. 1268–1293, thirdquarter 2015.
- [22] K. O. Bailey, J. S. Okolica, and G. L. Peterson, "User identification and authentication using multi-modal behavioral biometrics," *Comput. Secur.*, vol. 43, pp. 77–89, Jun. 2014.
- [23] N. T, "Did you know how many different kinds of sensors go inside a smartphone?," *Phone Arena*, 06-Jul-2014. [Online]. Available: http://www.phonearena.com/news/Did-you-know-how-many-different-kinds-of-sensors-go-inside-a-smartphone_id57885. [Accessed: 07-Mar-2017].
- [24] W.-H. Lee and R. Lee, "Multi-sensor authentication to improve smartphone security," presented at the International Conference on Information Systems Security and Privacy, 2015.
- [25] "Sensors Overview | Android Developers." [Online]. Available: https://developer.android.com/guide/topics/sensors/sensors_overview.html. [Accessed: 07-Mar-2017].
- [26] K. K. Rachuri, T. Hossmann, C. Mascolo, and S. Holden, "Beyond location check-ins: Exploring physical and soft sensing to augment social check-in apps," in *Pervasive Computing and Communications (PerCom), 2015 IEEE International Conference on*, 2015, pp. 123–130.
- [27] Z. Arnold, D. Larose, and E. Agu, "Smartphone Inference of Alcohol Consumption Levels from Gait," in *2015 International Conference on Healthcare Informatics*, 2015, pp. 417–426.
- [28] J.-K. Min, A. Doryab, J. Wiese, S. Amini, J. Zimmerman, and J. I. Hong, "Toss 'n' turn: smartphone as sleep and sleep quality detector," 2014, pp. 477–486.
- [29] "Android 7.0 Behavior Changes," *Android Developers*, 06-Mar-2017. [Online]. Available: <https://developer.android.com/about/versions/nougat/android-7.0-changes.html#doze>.
- [30] J. Hildenbrand, "Inside Marshmallow: What is Doze, how do I use it and what does it do?," *Android Central*, 14-Oct-2015. [Online]. Available: <http://www.androidcentral.com/inside-marshmallow-what-doze-how-do-i-use-it-and-what-does-it-do>. [Accessed: 18-Dec-2016].
- [31] "Optimizing for Doze and App Standby | Android Developers." [Online]. Available: <https://developer.android.com/training/monitoring-device-state/doze-standby.html>. [Accessed: 25-Apr-2017].
- [32] D. Siganos and C. Stergiou, "Neural Networks." [Online]. Available: https://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html#%20From%20Human%20Neurones%20to%20Artificial%20Neurones. [Accessed: 07-Mar-2017].
- [33] "neural_net2.jpeg (791×388)." [Online]. Available: http://cs231n.github.io/assets/nn1/neural_net2.jpeg. [Accessed: 07-Mar-2017].
- [34] F. van Veen, "The Neural Network Zoo," *The Asimov Institute*, 14-Sep-2016. .
- [35] R. V. Yampolskiy and V. Govindaraju, "Behavioural biometrics: a survey and classification," *Int. J. Biom.*, vol. 1, no. 1, pp. 81–113, 2008.
- [36] S. Buthpitiya, A. K. Dey, and M. Griss, "Soft authentication with low-cost signatures," in *Pervasive Computing and Communications (PerCom), 2014 IEEE International Conference on*, 2014, pp. 172–180.
- [37] D. Jurafsky and J. Martin, *Speech and Language Processing*. Stanford Lagunita, 2014.
- [38] R. Murmuria, A. Stavrou, D. Barbará, and D. Fleck, "Continuous Authentication on Mobile Devices Using Power Consumption, Touch Gestures and Physical Movement of Users," in *International Workshop on Recent Advances in Intrusion Detection*, 2015, pp. 405–424.
- [39] R. Murmuria and A. Stavrou, "Authentication Feature and Model Selection using Penalty Algorithms," in *Symposium on Usable Privacy and Security (SOUPS)*, 2016.
- [40] L. Ballard, F. Monroe, and D. P. Lopresti, "Biometric Authentication Revisited: Understanding the Impact of Wolves in Sheep's Clothing.," in *USENIX Security*, 2006.

- [41] A. Alzubaidi and J. Kalita, "Authentication of Smartphone Users Using Behavioral Biometrics," *IEEE Commun. Surv. Tutor.*, vol. 18, no. 3, pp. 1998–2026, thirdquarter 2016.
- [42] L. Fridman, S. Weber, R. Greenstadt, and M. Kam, "Active Authentication on Mobile Devices via Stylometry, Application Usage, Web Browsing, and GPS Location," *IEEE Syst. J.*, vol. PP, no. 99, pp. 1–9, 2016.
- [43] J. W. | M. 13, 2015, and 4:35 Am Pst, "How to set Android Lollipop Trusted places to bypass your screen lock," *TechRepublic*. [Online]. Available: <http://www.techrepublic.com/article/pro-tip-set-android-lollipop-trusted-places-to-bypass-your-screen-lock/>. [Accessed: 06-Mar-2017].
- [44] Gs. Labs, *GSam Battery Monitor*. GSam Labs, 2016.