IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 22, 2021, accepted November 28, 2021, date of publication December 3, 2021,
date of current version December 10, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3132767

Quad Swipe Pattern: A New Point-of-Entry Security Measure for Smartphone Users

JAMES F. BROWN, MD S. HOSSAIN, (Senior Member, IEEE), AND LISA LANCOR, (Member, IEEE)
Department of Computer Science, Southern Connecticut State University, New Haven, CT 06515, USA

Corresponding author: Md S. Hossain (hossainm3@southernct.edu)

This work involved human subjects or animals in its research. Approval of all ethical and experimental procedures and protocols was granted by the Southern Connecticut State University Institutional Review Board (IRB) under Approval No. 15-143.

ABSTRACT In this paper, we develop a new point-of-entry security measure for smartphone users. We devise a concept, the ''Quad Swipe Pattern'', which includes four swipes from a user in four directions and utilizes the user's swipe behavior for authentication. The Quad Swipe Pattern overcomes several shortcomings present in current point-of-entry security measures. We performed several experiments to demonstrate the effectiveness of the Quad Swipe Pattern in smartphone user authentication. We evaluated the Quad Swipe Pattern using five machine learning classifiers, three datasets of different sizes, and five different fingers. In addition, we studied how fusion of information from multiple fingers and multiple classifiers can improve the performance of Quad Swipe Pattern. All of our experimental results show significant promise of the Quad Swipe Pattern as a new point-of-entry security measure for smartphones. With a Neural Network model, the Quad Swipe Pattern achieves the Accuracy of 99.7%, False Acceptance Rate of 0.4%, and False Rejection Rate of 0%. With Support Vector Machine, the Quad Swipe Pattern achieves the Accuracy of 99.5%, False Acceptance Rate of 0.4%, and False Rejection Rate of 1.7%. With fusion of two best fingers, the Quad Swipe Pattern demonstrates an excellent performance of a zero Equal Error Rate.

INDEX TERMS Biometrics, authentication, smartphone security, touchscreen, swipe gesture.

I. INTRODUCTION

Smartphones continue to play a more important role in our lives with every passing day. Individuals use their phones for a wide variety of tasks that range from staying in touch with friends and family, checking a credit card balance, to reading the latest news article. Bank records, passwords, web usage, schedules, plans, interests, and personal relationships are all stored in smartphones. Unfortunately, this versatility in smartphones has generated its own risks. Study [1] reports that consolidation of vast amounts of information in one place is the major security concern in smartphones.

To secure data on smartphones, several point-of-entry security measures such as passwords, passcodes, and patterns currently exist. However, none of these measures are perfect; each of them has some limitations. One of the main issues with passwords and passcodes is that people often choose their entry codes for convenience, rather than security. Studies show that out of 204,508 iPhone passcodes, ten different passcodes made up 15% of the passcodes assessed [2]. The aftermath is that nearly 1 in 7 iPhones could be unlocked just by going through the 10 most common passcodes. Flaws in using a password is further demonstrated by Li *et al.* [3], where they analyzed over 100s of millions of leaked passwords from popular Chinese and English websites. They found that the most common password was ''123456''. Another common password they found was the word ''password''. This study demonstrates how smartphone users can allow their security to be compromised by choosing simple, guessable passwords.

Another issue with password/passcode involves the ''smudge attack'' [4]. When a smartphone user uses the phone, he/she leaves behind a trail of smudges on the screen. ''Smudge attacks'' occur when an unauthorized user looks at the smudges on a smartphone screen and then uses those smudges to guess a lock-screen passcode.

An alternative to a password/passcode based security protocol is a user-defined pattern, where a user needs to replicate their preset pattern to unlock the phone. Unfortunately,

The associate editor coordinating the review of this manuscript and approving it for publication was Zahid Akhtar.

researchers demonstrate grave flaws in pattern-based security. A study shows that it is significantly easier to steal a pattern compared to a six-digit PIN [5]. Out of 1,173 participants, 64% were able to reproduce a pattern containing six points. If participants were shown a pattern twice, 80% of those tested were able to reproduce the pattern. In comparison, with one viewing, 11% of participants could recall a six-digit PIN, and 27% could recall the PIN correctly after seeing the PIN twice.

In order to combat the stealing of passwords and patterns, researchers have developed biometric-based security protocols for smartphones. Rather than using something a person knows, a biometric-based security protocol relies on something a person has or is based on his/her behavior. Several smartphones are currently using physical biometrics such as fingerprint and face images. For example, iPhone's Touch ID uses fingerprint images in order to grant access to the phone. While Touch ID provides a good layer of security, it is not infallible. Article [6] highlights the flaws of Touch ID. The small sensor used by Apple can be compromised by dust, oils, and moisture. Apple protects its delicate sensor by covering it with a piece of crystal. Together, the small sensor and its crystal covering limit the amount of data that can be captured. This limited amount of data forced Apple to compromise on its False Acceptance and False Rejection Rates. Additionally, studies show that it is possible for fingerprints to be stolen and used to unlock a device [7]. Another physical biometric protocol, facial recognition, is also being used by several smartphones. However, face biometrics is still challenged by lighting conditions and face orientation when looking at a smartphone [8]. Simple things like leaning over too far or not looking directly at the camera can foil facial recognition software. These failures force users to rely on their passcodes and passwords in order to gain access to their smartphones.

A. OUR CONTRIBUTION AND NOVELTY

In this paper, we develop an authentication system for smartphone users, which resolves the above mentioned issues. Specifically, we devise a concept of a Quad Swipe Pattern which uses swipe behavior of a smartphone user for authentication. The Quad Swipe Pattern is a point-of-entry authentication measure. It includes four swipes from the smartphone user in four directions, one from left to right, one from right to left, one from up to down, and one from down to up. With each swipe from a user, behavioral information such as pressure, size, timing, position, etc. are captured and then combined to be used for authentication.

We performed several experiments to demonstrate the effectiveness of the Quad Swipe Pattern for smartphone user authentication. We evaluated the Quad Swipe Pattern using five machine learning classifiers: Support Vector Machine, Naive Bayes, two different models of Neural Networks, and Random Forest. We collected swipe data from 45 users in three different sessions. We created three different datasets to examine how Quad Swipe Pattern performs when the size of datasets and the number of training sessions vary. We experimented with five fingers to see the applicability of Quad Swipe Pattern on different fingers and different hands. In addition, we studied how fusion of information from multiple fingers and multiple classifiers can improve the authentication accuracy. All of our experimental results show significant promise of Quad Swipe Pattern as a new point-of-entry security measure for smartphones.

1) BENEFITS OF THE QUAD SWIPE PATTERN

The Quad Swipe Pattern is an important advancement in smartphone security gateways. It overcomes several shortcomings present in current point-of-entry security measures. First of all, it does not need any passcodes, PINs, or passwords. Not needing a passcode or password will eliminate user errors, like choosing an easy to guess or common password. The Quad Swipe Pattern is also immune to shoulder surfing and smudge attacks. An individual with malicious intent will not be able to gain anything from watching the Quad Swipe Pattern in action. The Quad Swipe Pattern also has advantages over facial recognition. Facial recognition requires adequate lighting and presentation of one's face to a camera. Unlike facial recognition protocol, the Quad Swipe Pattern will work in any conditions so long as proper contact can be made with the touchscreen. The Quad Swipe Pattern based authentication system does not require extra hardware such as an iris scanner or fingerprint scanner, which significantly can reduce the manufacturing cost of smartphones. Finally, the Quad Swipe Pattern uses information that would be very difficult to steal. By using biometric information that is harder to attain, the Quad Swipe Pattern gains another edge over the security gateways currently on the market. Millions of people swipe on their phones every day, and hence, the swipes required by the Quad Swipe Pattern will be an easy way to ensure the security of their information.

The remainder of the paper is organized in the following manner. Section II contains a review of literature related to our work. Section III contains a detailed description of the Quad Swipe Pattern and its implementation. Section IV covers data collection and data processing, while Section V discusses feature extraction. Section VI covers our experimental design, while Section VII covers the results of those experiments. Section VIII discusses our findings and finally, Section IX puts concluding remarks.

II. RELATED LITERATURE

Once past a phone's lock screen, a user's access to a smartphone's data is almost unlimited. Therefore, several research groups have been working towards improving smartphone security gateways. Modern touchscreens collect a variety of information and that information can be used to combat security breaches. The work of several researches demonstrate how touchscreen data collected from smartphones can be used to identify and authenticate individuals.

A. STUDIES ON TOUCH GESTURES

The measurement capabilities of a modern touchscreen are examined in detail by Wang *et al.* [9]. The authors explored human finger input properties. They collected a variety of touch data such as contact area, contact shape, and contact orientation from a subject's thumb, index, middle, ring, and little fingers. The index, ring, and middle fingers proved to be the best fingers from which to record data.

Feng *et al.* [10] demonstrated the ability to identify a user based on data collected from a touchscreen. The researchers collected touch features such as touch location, swipe speed, touch size, and the time between touch events. All of this information was collected in the background during normal usage of an Android device. The collected information was used by a Decision Tree classifier to authenticate individuals. A True Positive Rate of 91% and a True Negative Rate of 93% was achieved.

Rilvan *et al.* [11] used capacitive data collected from touchscreen to authenticate smartphone users. They demonstrated significant promise of machine learning classifiers and principal component analysis in user authentication. They achieved 98.27% accuracy with Support Vector Machine (SVM) and 97.78% accuracy with Random Forest. Guo *et al.* [12] also experimented with capacitive touch data for user authentication. Using SVM, they achieved a False Acceptance Rate of 0.1% and a False Rejection Rate of 5.5%. Their work further demonstrates how SVM is an effective classifier when used with touch data.

Further evidence for the use of touchscreen data and machine learning classifiers comes from the work of Jain *et al.* [13]. Using an Android phone, they measured the touch locations and pressures and obtained an Equal Error Rate (EER) of 3.5% using SVM. Zheng *et al.* [14] corroborates the work of Jain *et al.* in their study and demonstrates how tap behavior on a smartphone's touch screen can be used to authenticate users. Using the features of acceleration, duration, pressure and size of a tap they achieved an EER of 3.6%.

Coakley *et al.* [15] also demonstrated the effectiveness of tap gestures for user authentication in smartphones. They studied a 10-digit PIN entered by 52 users and achieved an EER of 3.9%. Chang *et al.* [16] experimented with 6, 8, and 10-digit PINs and observed a significant amount of reduction in EER going from the shortest to the longest PIN. Buschek *et al.* [17] experimented with 6 different PINs, where data was collected from 28 participants over two weeks, and obtained promising results using spatial touch features. Another study by Buschek *et al.* [18] compared the performance of an index finger and thumb by collecting tap data from 24 participants and found thumb as the better performer.

Ku *et al.* [19] explored open lock patterns for smartphone user authentication. With an open lock pattern, the pattern is displayed and the user log in by dragging his/her fingers over a 3×3 grid of dots following the displayed pattern. Because the pattern is open (displayed), no memorization is required by the users. They achieved an EER of 2.66%. Similar to Ku *et al.*, Haberfeld *et al.* [20] devised a concept of an "open code" biometric tap pad to authenticate users in smartphones, which eliminates the need of memorizing PINs. They achieved an EER of 13.56%.

Leran *et al.* [21] focused solely on zoom-in and zoom-out gestures for user authentication and identification in smartphones. They extracted a rich set features from zoom gestures and obtained an authentication performance of EER 10.6% using SVM classifier. They achieved the best identification performance of accuracy 65.5% accuracy, precision 69.6% precision, and recall 67.9% using the random forest (RF) classifier.

B. STUDIES ON SWIPE GESTURES

Because our work uses swipe gestures for user authentication, below we describe the works focusing swipe gestures.

De Luca *et al.* [22] is one of the earliest works utilizing swipe gestures for user authentication. They collected swipe data from 48 participants using multiple swipe patterns. They applied the Dynamic Time Warping algorithm [23] for classification. Using pressure, swipe coordinates, and touch size features they achieved an accuracy of 77%.

SaeBae *et al.* [24] demonstrated the plausibility of using multiple simultaneous swipes on a touchscreen to identify a user. A multi-touch gesture was recorded on an iPad's touchscreen which consisted of placing all five finger tips on the screen and moving them in specific ways. Movements included pivoting the four fingers around the thumb in both the clockwise and counter clockwise direction, as well as dragging all fingertips across the screen. Twenty-two multi-touch gestures from 34 users were tested for user authentication. Using a single multi-touch gesture, they achieved an EER of 10% and using double multi-touch gestures, they achived an EER of 5%.

Frank *et al.* [25] experimented with horizontal and vertical swipes, where data was collected from 41 users. They extracted 27 features from each swipe and performed classification using K-nearest neighbors (KNN) and SVM. Using a single swipe gesture, they achieved an EER of 13% and using 12 swipe gestures, they achieved an EER of 4%.

Shahzad *et al.* [26] demonstrated a successful use of gestures for the secure unlocking of touch screen devices. The authors designed 39 simple gestures that were easy to perform and finally selected 10 most effective gestures. They performed experiments by collecting a set of 15009 training samples from 50 users. Using three gestures, they achieved and EER of 0.5%.

Antal *et al.* [27] worked with horizontal swipes for user authentication on phones. They recorded micro movements during swipes and extracted a rich set of features. They achieved an EER of 4% using a single swipe gesture.

Kumar *et al.* [28] developed an authentication system for smartphone users by fusing information from typing, swiping, and phone movement patterns. They performed

experiments on touchscreen data collected from 28 users and achieved an accuracy of 93.33% using feature-level fusion of swiping and phone movement patterns.

Siirtola *et al.* [29] studied context-based swipe gestures for continuous user authentication in smartphones. The users performed the swipes within different contexts such as while reading, navigating, sitting or walking. They observed that the performance of swipe gestures were significantly affected by the differing contexts.

Rilvan *et al.* [30] used capacitive swipes for user athentication and identification in smartphones. A capacitive swipe is defined as a series of capacitive frames extracted from the user's swipe. After preprocessing the capacitive frames, they applied principal component analysis for feature extraction and SVM for classification. They achieved an accuracy of 79.88%.

C. OUR STUDY

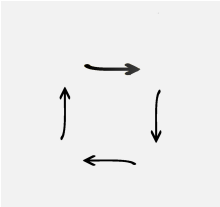
While much research has been conducted to explore smartphone gestures and swipes, our work differs from the previous work in several ways. The biggest difference between our Quad Swipe Pattern and the previously mentioned research is our intent to design an easy to use smartphone security gateway. Most of the research discussed previously explores the plausibility of authenticating an individual continuously. Unlike these studies, the Quad Swipe Pattern is designed to be used as a gateway security protocol rather than a continuous authentication protocol. We find three works, specifically, Ku *et al.* [19], Haberfeld *et al.* [20], and Shahzad *et al.* [26] who explore point-of-entry authentication using touch gestures like ours and eliminate the need for memorizing password, PIN, or passcode. Ku *et al.* [19] explore open lock patterns, Haberfeld *et al.* [20] explore open passcodes, and Shahzad *et al.* [26] explore custom gestures. Our experimental results show that the Quad Swipe Pattern achieves a better performance than these three studies. The Quad Swipe Pattern relies on using a series of four swipes in four directions, which allows collecting more information to make a better authentication decision.

The Quad Swipe Pattern advances the state-of-the-art in other ways too. We perform rigorous experiments to learn the effectiveness of different fingers in user authentication. Specifically, we explore five different fingers from two hands and fusion of multiple fingers. We did not find any previous work that explored different fingers so rigorously. Additionally, we define a rich set of features, explore five machine learning classifiers, and fusion of multiple classifiers.

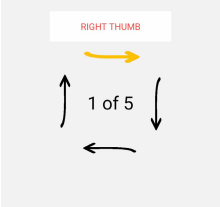
III. QUAD SWIPE PATTERN

A. BASICS

Figure 1 (a) shows the Quad Swipe Pattern we designed for user authentication in smartphones. The Quad Swipe Pattern includes four swipes in four directions. The four swipes recorded are from left to right, top to bottom, right to left, and bottom to top. The Quad Swipe Pattern is designed to be simple and easy to use. The slight curve in the guiding arrows



(a) The Quad Swipe Pattern concept.



(b) The Quad Swipe Pattern used in our Android application for data collection.

FIGURE 1. Quad Swipe Pattern: (a) shows the concept and (b) shows when it is used in our Android application for data collection. In (b), the red "RIGHT THUMB" text informs a participant of what finger to use, while the yellow arrow indicates a participant's next swipe. The "1 of 5" in the center of the arrows indicates which round of swiping a participant is on. In this screenshot, the user is on the first swipe of the first round.

makes it quicker and easier for users to transition from one swipe to the next. Swiping around a set of arrows is very easy and mimics swiping motions often used throughout the day. The Quad Swipe Pattern uses the center of the screen so that the users are comfortable with the swiping.

A key benefit of the Quad Swipe Pattern is its creation of a comprehensive swipe profile. The design of Quad Swipe Pattern allows collecting touch data on different portions of the smartphone screen. By collecting data from different sections of the phone-screen, we are able to build a complete swipe profile for a user. For example, a right handed individual may interact differently with the right portion of the screen than the left, or a person may swipe differently when swiping up or down. Overall, the Quad Swipe Pattern is designed to be simple and easy to use, while also building a comprehensive swiping profile of an individual. The combination of these two factors allows the system to quickly capture data and compare it to a user's comprehensive profile.

B. IMPLEMENTATION

To implement the Quad Swipe Pattern and collect data from smartphone users, we developed an Android application. Fig. 1(b) shows the Quad Swipe Pattern used in our Android application. The application required the participants to swipe clockwise around a square pattern of arrows. A participant would make four swipes on the screen following an arrow highlighted in yellow. One swipe across the top of the screen to the right, one down the right side of the screen, one to the left across the bottom of the screen, and finally one swipe up across the left edge of the screen. To guide participants

through the data collection, the arrow of current swipe was highlighted in yellow as the user swiped through the Quad Swipe Pattern. During these swipes, biometric swipe data were collected from each participant. The coordinates of a swipe were recorded along with the pressure on the screen, the duration of the swipe gesture, and the size of touches during a swipe. One set of fours swipes was considered to be one round. Each participant was required to complete five rounds of swipes for each finger. Data were collected from five fingers of a participant: right index finger, right thumb, left index finger, left thumb, and the middle finger on a participant's self-identified dominant hand. In total, 25 rounds of swipe data were collected from one participant during a session. Multiple directions and rounds were used to build a more comprehensive profile of a participant's swipes. Multiple fingers were tested to see how different fingers perform on Quad Swipe Pattern.

IV. DATA

A. DATA COLLECTION

The participants in our experiment were students, faculty, and staff at Southern Connecticut State University. None of the participants were compensated for their time and there was no risk to the participants since they were interacting with a smartphone's touchscreen in a normal manner. Nexus 5 smartphones were used for data collection. Our Android application was deployed and run on each device in order for participants to use the Quad Swipe Pattern. The participants needed approximately 5 minutes to complete the data collection procedure. We collected data in three sessions. In each session, a participant gave 25 rounds of data. In total, 45 participants gave their swipe data. All 45 participants gave data at least in one session, 15 participants gave data at least in two sessions, and 10 participants gave data in all three sessions.

B. DATA PREPROCESSING

In order to prepare the data for training the machine learning classifiers and testing, the data were cleansed by removing irregularities. In a session, a participant gave five rounds of swipes by each finger. In each round, the participant swiped in four directions. So, for each finger, a session should ideally have five swipes in each direction. However, sometimes exactly five swipes were not recorded in each direction. Sometimes extraneous swipes were recorded in some directions. Also sometimes a swipe was missing in some directions. We handled this irregularities very carefully. If more than five swipes were recorded in some direction, we accepted the first five swipes and removed the others in that direction. If less than five swipes were recorded in some direction, we estimated the features of the missing swipe by averaging the corresponding feature values of the recoded swipes in that direction. In addition to eliminating irregularities, we removed outliers from the feature values. We used Gaussian distribution technique for outlier detection.

TABLE 1. List of features extracted from a swipe.

Pressure Features	**Velocity Features**
Start Point Pressure	Average Swipe Velocity
End Point Pressure	Maximum Swipe Velocity
Maximum Pressure	Minimum X Velocity
Minimum Pressure	Minimum Y Velocity
Average Pressure	Average X Velocity
Size Features	Average Y Velocity
Start Point Size	Maximum X Velocity
End Point Size	Maximum Y Velocity
Maximum Touch Size	**Time Features**
Minimum Touch Size	Time from Start to Max Pressure
Average Touch Size	Time from Start to End Point
Coordinate Features	**Distance Features**
Maximum X Coordinate	Distance from Start to Max Pressure
Maximum Y Coordinate	Distance from Start to End Point
Minimum X Coordinate	Swipe Arc Length
Minimum Y Coordinate	**Other Features**
Average X Coordinate	Product of Size and Time
Average Y Coordinate	Product of Max Size and Pressure
Slope from Start to End Point	Swipe Momentum
Slope from Start to Mid Point	

V. FEATURE EXTRACTION

We extracted 34 features from a swipe in one direction. A round of swipes consists of four swipes in four directions. Hence, we extracted 34*4 (which is equal to 136) features from one round of swipes, which were fed into the machine learning classifiers for training and testing. Table 1 shows the list of features we used in our study. Below, we briefly explain these features.

1) PRESSURE FEATURES

Pressure features record how much force a participant imparts on the screen during a swipe. The ''Start Point Pressure'' feature records the force of the touch at the beginning of the swipe. The ''End Point Pressure'' feature measures the pressure on the screen at the end of the user's swipe. The ''Maximum Pressure'' feature measures the greatest amount of pressure a participant exerts on the screen during a swipe. The ''Minimum Pressure'' feature measures the smallest amount of pressure a participant imparts on the screen. The ''Average Pressure'' feature sums all of the pressures recorded during a swipe and then finds the mean. The Pressure values in Nexus devices range from 0 to 1, with 0 being no pressure and 1 being the maximum pressure.

2) SIZE FEATURES

The size of a touch during swiping is recorded by the number of pixels activated by the touch event. The ''Start Point Size'' feature records the size of the touch at the beginning of a swipe. The ''End Point Size'' feature records the size of the touch at the end of a swipe. The ''Maximum Touch Size'' feature records the largest area touched during a swipe. Similarly, the ''Minimum Touch Size'' feature records the smallest area touched during a swipe. The ''Average Touch Size'' feature represents the average touch size during a swipe.

3) COORDINATE FEATURES

These features record the touch screen coordinates activated during a swipe. The coordinates are measured on a screen

where the origin, $(0, 0)$, is located in the top left corner of the screen, the X-axis runs horizontally along the top of the screen, and the Y-axis runs vertically along the left edge of the screen. The X-coordinates increase approaching the right edge of the device's screen and the Y-coordinates increase approaching the bottom edge of the screen. The ''Maximum X Coordinate'' feature records the largest X-coordinate touched during a swipe, while the ''Maximum Y Coordinate'' feature records the largest Y-coordinate touched during a swipe. The ''Minimum X Coordinate'' feature records the smallest X-coordinate touched during a swipe. The ''Minimum Y Coordinate'' feature records the smallest Y-coordinate touched during a swipe. The ''Average X Coordinate'' and ''Average Y Coordinate'' features represent the mean of all X and Y coordinates, respectively, touched during a swipe.

We define two additional features based on the coordinates of the screen. The ''Slope from Start to End Point'' and ''Slope from Start to Mid Point'' features allow us to compare how coordinates change over the course of a swipe.

4) VELOCITY FEATURES

Velocity features are measured in pixels per millisecond. The ''Average Swipe Velocity'' feature represents the average velocity of a swipe. The greatest instantaneous velocity of a swipe is recorded as the ''Maximum Swipe Velocity''. Swipe velocity is also broken down into X and Y components. The ''Minimum X Velocity'' records the slowest instantaneous X-component of velocity for a swipe. The ''Minimum Y Velocity'' records the slowest instantaneous Y-component of velocity for a swipe. The ''Average X Velocity'' records the average X-component of velocity for a swipe. The ''Average Y Velocity'' records the average Y-component of velocity for a swipe. The ''Maximum X Velocity'' records the greatest instantaneous X-component of velocity for a swipe. The ''Maximum Y Velocity'' records the greatest instantaneous Y-component of velocity for a swipe.

5) TIME FEATURES

The ''Time from Start to Max Pressure'' feature records the time that elapses between the beginning of a participant's swipe and the point at which the maximum pressure is recorded. The goal of this feature is to identify the time of the swipe at which a participant puts the most force on the screen. The ''Time from Start Point to End Point'' feature records the length of time it takes for a participant to complete his or her swipe.

6) DISTANCE FEATURES

The ''Distance from Start to Max Pressure'' feature provides the Euclidean distance from the start of a participant's swipe to the point at which the maximum pressure is recorded. The ''Distance from Start to End Point'' feature provides the Euclidean distance between the start and end points of a swipe. The ''Swipe Arc Length'' feature provides the arc length of a swipe.

7) OTHER FEATURES

We define three more features which combine the pressure, size, and time. The ''Product of Size and Time'' feature is calculated by multiplying the total area touched with the duration (time) of the swipe. The ''Product of Max Size and Pressure'' feature is calculated by multiplying the maximum size with the maximum pressure. The ''Swipe Momentum'' feature is calculated by multiplying the sum of all pressures with the duration (time) of the swipe.

After the features are extracted, we applied the z-score normalization technique to scale all of the features.

VI. EXPERIMENTS

A. AUTHENTICATION PROCEDURE

The goal of the Quad Swipe Pattern is to provide a reliable and secure method for user authentication. User authentication in a smartphone is the process of determining whether a person has clearance to use the phone. An authorized user is allowed to use the device, while an unauthorized user is rejected and barred access. Our authentication procedure is implemented using a binary classifier. The classifier is trained using labeled samples. The training samples are either labeled ''genuine'' for an authorized user or ''impostor'' for an unauthorized user. After training, the classifier is tested with a set of unlabeled samples. In our experiments, a classifier generates two different types of decisions: (1) a *binary decision*, where a test sample is either labeled as ''genuine'' or an ''impostor'' and (2) a *confidence value*, which is a dissimilarity score and ranges from 0 to 1. The closer the score is to 0, the more certain the classifier is that the test sample belongs to a genuine person.

B. DATA DIVISION

We have divided our collected data into three sets: Dataset 1, Dataset 2, and Dataset 3. These three datasets are used to evaluate how the amount of training data affects the performance of Quad Swipe Pattern. It was anticipated that more training data per participant would allow for the construction of a more complete swipe profile. A better swipe profile would allow for improved performance.

Dataset 1 contains the swipe data collected from 45 participants, where each participant gave data in one session. Dataset 2 contains the swipe data collected from 15 participants, where each participant gave data in two sessions. Finally, Dataset 3 contains the swipe data collected from 10 participants, where each participant gave data in three sessions. In a session, a participant gave 5 rounds of swipe data for each finger we experimented with. Every Dataset (1, 2, and 3) was divided into training and testing sets. Three rounds of swipe data for each finger, from each session, were used for training and the rest two rounds of data were used for testing. As a result, the training part of Dataset 1 has three rounds of swipe data for each finger per person, the training part of Dataset 2 has six rounds of data for each finger per person, and finally, the training part of Dataset 3 has 9 rounds

TABLE 2. Dataset statisticcs.

Dataset	Number of Partici-pants	Number of Sessions	Training Samples (Number of Rounds per Finger)	Testing Samples (Number of Rounds per Finger)
1	45	1	45*3	45*2
2	15	2	15*6	15*4
3	10	3	10*9	10*6

of swipe data for each finger per person. Table 2 shows a summary of our data division. Note that Dataset 1 contains the maximum number of participants, however, the smallest amount of training data for each participant. In contrast, Dataset 2 and 3 contain a smaller number of participants, but more training data for each participant.

C. MACHINE LEARNING CLASSIFIERS

We performed our experiments with five machine learning classifiers: Support Vector Machine (SVM), Naive Bayes (NB), two different models of Neural Networks: Neural Network-3 (NN-3) and Neural Network-28 (NN-28), and Random Forest (RF). Below, we briefly explain these classifiers.

1) SUPPORT VECTOR MACHINE

The Support Vector Machine (SVM) classifier works by creating a set of hyperplanes that separate data into different classes [31]. A good hyperplane is characterized by having a decent amount of separation between self and the nearest point of training data. SVM is effective at generalizing and is a memory efficient algorithm. Because of the limited amount of computing power available in a smartphone, it was critical to explore this memory efficient algorithm on our data. After performing preliminary experiments, we chose linear kernel to implement the SVM.

2) NAIVE BAYES

The Naive Bayes (NB) classifier works by assuming that all of the features are independent [32]. We considered exploring Naive Bayes in our study because it works very fast, which is an expected feature of a point-of-entry authentication protocol in smartphones. The Naive Bayes classifier is implemented using scikit-learn's default settings which uses an alpha value of 1.0 [33], [34].

3) NEURAL NETWORKS

Neural Networks (NN) are modeled after connections made in the human brain. The network works by taking data from the input layer, and then passing it through one hidden layer or a series of hidden layers which is responsible for generating the output. The nodes in one hidden layer make connections with nodes in other layers or with the input layer. These connections between layers are weighted and the weights determine what information is used to make a classification decision. In the final layer of the network, the output is presented. In our experiments, we modeled two Neural Networks: (1) Neural Network 3 (NN-3), which is light weight and (2) Neural Network 28 (NN-28), which is heavy weight.

The NN-3 is our light weight neural network because it has only one hidden layer with three nodes. This neural network was configured in this manner due to its fast training and testing times, an important characteristic given the hardware resources of a smartphone. While our light weight neural network contains only three nodes in its sole hidden layer, the heavy weight neural network, NN-28, has 28 nodes per hidden layer. NN-28 has 8 hidden layers, seven more than NN-3. This neural network was configured in this manner to explore how more nodes and layers would affect the performance of a neural network. The heavy weight network also allows us to evaluate if the speed penalty of the more complex network is a worthwhile trade-off.

4) RANDOM FOREST

The Random Forest (RF) classifier creates a multitude of decision trees in order to make its classification decision [35]. The decisions returned by all of these trees are then combined to produce a final decision. We wanted to examine how our Quad Swipe Pattern performs with an ensemble learning method. As an ensemble method, we chose RF because of its promising performance in a previous study [11].

D. FUSION

In our experiments, we also examined how Quad Swipe Pattern performs when information from two best performing fingers are fused, and also, when information from five classifiers are fused. The fusion of two best performing fingers, for each classifier, was accomplished by taking the average of the individual feature values of the two fingers. The two best preforming fingers differed for each dataset and for each classifier. When the classifier generated a binary decision (genuine/impostor), the best two fingers were chosen based on the accuracy. When the classifier generated a confidence value, the best two fingers were chosen based on the EER.

We fused the information from five classifiers at the decision or score level. When the classifiers generated binary decisions (genuine/impostor), we applied majority voting to make a final decision. When the classifiers generated confidence values, we averaged those values to estimate a fused score.

E. EVALUATION METRICS

1) ACCURACY, FAR, AND FRR

When a binary decision was requested from the classifier, a confusion matrix [36] was constructed. A confusion matrix gives: True Positive (number of correctly labeled genuines), True Negative (number of correctly labeled impostors), False Positive (number of incorrectly labeled impostors), and False

Negative (number of incorrectly labeled genuines). Several performance metrics were derived from the confusion matrix. The Accuracy is the percentage of correct predictions. The False Positive Rate (FPR) is the percentage of impostors which were incorrectly labeled as genuine. This is also referred to as the False Acceptance Rate (FAR). The False Negative Rate (FNR) is the percentage of genuine samples which were incorrectly labeled as impostors. This is also called the False Rejection Rate (FRR).

2) DET CURVE

The Detection Error Trade-off (DET) curve [37] plots the False Acceptance Rate (FAR) and False Rejection Rate (FRR) as the decision threshold goes from zero to one. To generate this curve, confidence values were obtained from the classifier. DET curves are useful when the testing set does not contain approximately equal numbers within the two classes, which was the case in our study. The DET curve can be used to determine another metric, the Equal Error Rate (EER). This is one of the most commonly reported metrics, and represents the point where the FAR is equal to the FRR. A perfect biometric system in terms of accuracy would have an EER of zero.

VII. RESULTS AND ANALYSIS

A. ACCURACY, FAR, AND FRR

Tables 3, 4, and 5 show the Accuracy, FAR, and FRR obtained by five machine learning classifiers and five fingers, where the experiment was performed on Dataset 1, 2, and 3, respectively. The rightmost three columns (labelled ''Fusion of Best Two Fingers'') in each table show the performance of the fusion of two best fingers. For each classifier, the best two fingers were selected based on Accuracy. The Accuracy values of the two best fingers which were used in fusion are marked in red in the tables. For example, in Table 3, Left Thumb and Middle finger were fused for SVM and Left Index finger and Left Thumb were fused for Naive Bayes. The bottom row in the table shows the performance of Majority Voting applied on five classifiers.

1) PERFORMANCE OF INDIVIDUAL CLASSIFIERS (NON-FUSION)

When we consider the performance of individual classifiers without any fusion (i.e., no fusion of fingers and no fusion of classifiers), Neural Network 28 (NN-28) and Support Vector Machine (SVM) are the two best classifiers based on the results reported in Table 3, 4, and 5. For Dataset 1 (see Table 3), NN-28 gives better Accuracy and FAR than SVM for almost all fingers (for some fingers, Accuracy values are the same). However, SVM outperforms NN-28 in terms of FRR for all fingers except the Left Index finger. For Datasets 2 and 3 (see Table 4 and 5), NN-28 outperforms SVM in terms of all three metrics Accuracy, FAR, and FRR for almost all fingers. Both NN-28 and SVM achieve their best performance with Middle finger and Dataset 3, where NN-28 achieves Accuracy 99.7%, FAR 0.4%, FRR 0% and SVM achieves Accuracy 99.5%, FAR 0.4%, FRR 1.7%.

Random Forest (RF) and Neural Network 3 (NN-3) achieve very good Accuracy and FAR, however, give very high FRR which is not acceptable in smartphone settings considering user convenience. Similarly, Naive Bayes is also not a good fit because of its poor performance in all combinations of fingers and datasets.

2) PERFORMANCE OF FUSION

When we fuse information from two best fingers, we see significant improvement in FAR in all datasets. For example, for Dataset 3, SVM gives FAR of 0.9% with Right Thumb and 0.4% with Middle finger. Fusion of these two fingers produces zero FAR. However, this improvement in FAR comes at the expense of higher FRR. Though FRR values are high in Datasets 1 and 2, they are tolerable in Dataset 3 which has more samples in the training set. For Dataset 3, Neural Network 28 and SVM give outstanding performance with fusion of Right Thumb and Middle finger. Neural Network 28 achieves Accuracy 99.5%, FAR 0%, and FRR 5%. Support vector machine (SVM) achieves Accuracy 99.2%, FAR 0%, and FRR 8.3%. Majority voting of five classifiers also improves the FAR at the expense of higher FRR, when compared with the best individual classifier. Dataset 3 gives tolerable FRR values. Two best performances by majority voting for Dataset 3 are Accuracy 99.3%, FAR 0.2%, FRR 5% with Right Thumb and Accuracy 98.8%, FAR 0.2%, FRR 10% with Middle finger. If we compare fusion of two best fingers with majority voting of five classifiers, we see that fusion of two best fingers provides more security with zero FAR.

3) WHICH FINGER GIVES THE BEST RESULT?

In Tables 3, 4, and 5, the Accuracy values of two best fingers are marked in red. We see that Middle finger continuously gives outstanding performance almost for all classifiers and in all three datasets. Generally people do not use their Middle finger for swipes because of its position. The Middle finger is located between the Index finger and Ring finger, which makes it less comfortable for swiping. Perhaps this phenomenon gives high variability to the Middle finger when it is used for swiping.

4) IMPACT OF USING MORE TRAINING SAMPLES

It is evident from Tables 3, 4, and 5 that the overall performance of the classifiers improves when more training samples are used to create a user profile. Specially, we observe significant improvement of FRR from Dataset 1 to 2 and from Dataset 2 to 3. Recall that Dataset 1 uses data collected in a single session, Dataset 2 uses data collected in two sessions, and Dataset 3 uses data collected in three sessions.

B. DET CURVES

Figures 2, 3, and 4 show the DET curves obtained by five machine learning classifiers, where the experiment

TABLE 3. Accuracy, FAR, and FRR obtained by five machine learning classifiers and five fingers, where the experiment was performed on Dataset 1. The rightmost three columns (labelled "Fusion of best two fingers") show the performance of the fusion of the two best fingers. For each classifier, the best two fingers were selected based on accuracy. The accuracy values of the two best fingers which were used in fusion are marked in red. For example, left thumb and middle finger were fused for SVM. The bottom row shows the performance of majority voting applied on five classifiers.

	R. Index			R. Thumb			L. Index			L. Thumb			Middle			Fusion of Best Two Fingers		
Classifier	Acc	FAR	FRR	Acc	FAR	FRR	Acc	FAR	FRR	Acc	FAR	FRR	Acc	FAR	FRR	Acc	FAR	FRR
SVM	98.2%	0.9%	44.4%	98.4%	0.9%	32.2%	98.1%	0.8%	51.1%	98.4%	0.8%	32.2%	98.5%	0.7%	38.9%	98.9%	0.003%	54.4%
Naive Bayes	87.7%	11.0%	68.8%	78.2%	21.1%	53.3%	88.8%	9.8%	68.9%	89.9%	8.7%	74.4%	88.4%	10.5%	62.2%	93.7%	4.6%	83.3%
NN-3	97.9%	0.1%	90.0%	97.9%	0.2%	86.7%	98.3%	0.28%	65.6%	98.0%	0.1%	83.3%	97.8%	0.4%	82.2%	98.0%	0.0%	90.0%
NN-28	98.5%	0.3%	55.6%	98.4%	0.7%	42.2%	98.7%	0.38	42.2%	98.5%	0.4%	47.8%	98.5%	0.5%	46.7%	98.5%	0.1%	68.9%
Random Forest	97.9%	0.1%	93.3%	98.0%	0.0%	92.2%	98.0%	0.03%	91.1	98.0%	0.1%	92.2%	98.0%	0.1%	82.2%	97.9%	0.0%	95.6%
Majority Voting	98.1%	0.1%	81.1%	98.3%	0.2%	66.7%	98.4%	0.1%	67.8%	98.2%	0.1%	74.4%	98.4%	0.1%	65.6%	NA	NA	NA

TABLE 4. Accuracy, FAR, and FRR obtained by five machine learning classifiers and five fingers, where the experiment was performed on Dataset 2.

	R. Index			R. Thumb			L. Index			L. Thumb			Middle			Fusion of Best Two Fingers		
Classifier	Acc	FAR	FRR	Acc	FAR	FRR	Acc	FAR	FRR	Acc	FAR	FRR	Acc	FAR	FRR	Acc	FAR	FRR
SVM	96.8%	1.8%	23.3%	95.7%	2.4%	31.7%	96.2%	2.0%	28.3%	97.0%	2.0%	16.7%	97.6%	0.1%	23.3%	97.6%	0.1%	35.0%
Naive Bayes	69.2%	32.0%	13.3%	69.0%	31.7%	21.7%	79.3%	20.1%	28.3%	62.7%	39.0%	13.3%	74.4%	25.7%	23.3%	85.4%	13.2%	33.3%
NN-3	95.2%	1.2%	55.0%	94.4%	1.2%	66.7%	94.8%	1.8%	51.7%	93.7%	0.4%	90.0%	95.8%	1.5%	41.7%	95.6%0	0.1%	65.0%
NN-28	97.4%	1.2%	21.7%	96.9%	1.2%	30.0%	96.7%	1.9%	23.3%	98.1%	0.5%	21.7%	97.6%	1.0%	23.3%	97.2%	0.1%	40.0%
Random Forest	94.0%	2.1%	60.0%	95.5%	0.5%	60.0	95.4%	0.6%	60.0%	95.1%	0.4%	68.3%	95.8%	0.0%	61.7%	94.6%	0.0%	81.7%
Majority Voting	97.4%	0.8%	26.7%	96.6%	0.6%	43.3%	96.3%	1.3%	36.7%	97.4%	0.6%	30.0%	97.3%	0.7%	30.0%	NA	NA	NA

TABLE 5. Accuracy, FAR, and FRR obtained by five machine learning classifiers and five fingers, where the experiment was performed on Dataset 3.

	R. Index			R. Thumb			L. Index			L. Thumb			Middle			Fusion of Best Two Fingers		
Classifier	Acc	FAR	FRR	Acc	FAR	FRR	Acc	FAR	FRR	Acc	FAR	FRR	Acc	FAR	FRR	Acc	FAR	FRR
SVM	97.7%	1.5%	10.0%	98.5%	0.9%	6.7%	96.7%	2.2%	16.7%	96.7%	2.6%	10.0%	99.5%	0.4%	1.7%	99.2%	0.0%	8.3%
Naive Bayes	87.7%	9.8%	35.0%	64.5%	38.3%	10.0%	87.0%	8.8%	50.0%	74.3%	27.0%	13.3%	84.7%	12.2%	43.3%	89.8	5.4%%	53.3%
NN-3	97.3%	2.0%	8.3%	96.3%	1.1%	26.7%	96.2%	1.7%	23.2%	95.0%	1.1%	40.0%	97.5%	1.5%	11.7	98.2%	0.0%	18.3%
NN-28	98.3%	1.1%	6.7%	99.2%	0.4%	5.0%	97.8%	1.5%	8.3%	98.7%	1.1%	3.3%	99.7%	0.4%	0.0%	99.5%	0.0%	5.0%
Random Forest	95.3%	0.0%	46.7%	95.5%	0.2%	43.3%	94.0%	0.6%	55.0%	94.5%	0.7%	48.3%	95.7%	0.6%	38.3%	94.0%	0.0%	60.0%
Majority Voting	98.5%	0.7%	8.3%	99.3%	0.2%	5.0%	96.8%	1.1%	21.7%	97.5%	0.9%	16.7%	98.8%	0.2%	10.0%	NA	NA	NA

was performed on Dataset 1, 2, and 3, respectively. Subfigures (a)-(e) in each figure include six DET curves: five curves generated from five fingers and one curve generated from the fusion of best two fingers. For each classifier, the best two fingers were selected based on EER. Subfigure (f) in each figure shows five DET curves corresponding to five fingers, each curve generated from the fusion of five classifiers. In each of the subfigures, there is a black dotted line that runs through the DET curves. This is the EER line and each DET curve's EER can be found upon it. Our observations from Figures 2, 3, and 4 are given below.

(1) Considering individual classifier performance (without any fusion of fingers or classifiers), SVM and Neural Network 28 (NN-28) consistently outperform the other three classifiers in all three datasets. Both SVM and NN-28 achieve near perfect EER of 0.0 with Middle finger on Dataset 3. Random Forest (RF) achieves some good performance across all three datasets, whereas, Neural Network 3 (NN-3) achieves good performance only for Dataset 3.

(2) Fusion of two best fingers significantly improves the performance of all classifiers in all datasets. For Dataset 3 (see Figure 4), we achieve near perfect EER of 0.0 with SVM and NN-28. The impact of the fusion of the two best fingers is more noticeable in Dataset 1 and 2 (see Figures 2 and 3), where the performance of an individual classifier is not perfect. Fusion of five classifiers using the average of the scores does not demonstrate significant performance improvement when compared with the best performing individual classifiers.

(3) In Datasets 1 and 2, the Middle finger does not always demonstrate the best performance; however, in Dataset 3, where more training samples per participant are used, the Middle finger consistently demonstrates the best performance. This provides strong evidence that when more samples are used to create a user profile, in addition to giving the best performance, it also gives consistent performance.

C. TIME ELAPSED FOR TRAINING AND TESTING

Table 6 presents the training and testing times required for each of the five classifiers. We recorded the timing information only for dataset 1. Also, for non-fusion experiments, we recorded the timing information only for middle finger experiments. Overall, the fastest classifier to train was the Naive Bayes. It took only 43 milliseconds. Naive Bayes was also the second fastest at evaluating the unseen dataset with a testing time of 5 milliseconds. After Naive Bayes, SVM was the second fastest classifier to train with a time requirement of 143 milliseconds. The testing time for SVM was 18 milliseconds, which was 3 milliseconds higher than the testing time for NN-28. Though NN-28 was faster than SVM in testing, it was almost 13 times slower than SVM in training. NN-3 was faster than NN-28 in both training and testing. Random Forest appeared as the slowest individual classifier in both training and testing.

The experiment that took the longest period of time to complete was majority voting. Majority voting took 7192 milliseconds in training and 416 milliseconds in testing. The time required by the fusion of the best two fingers fell between the

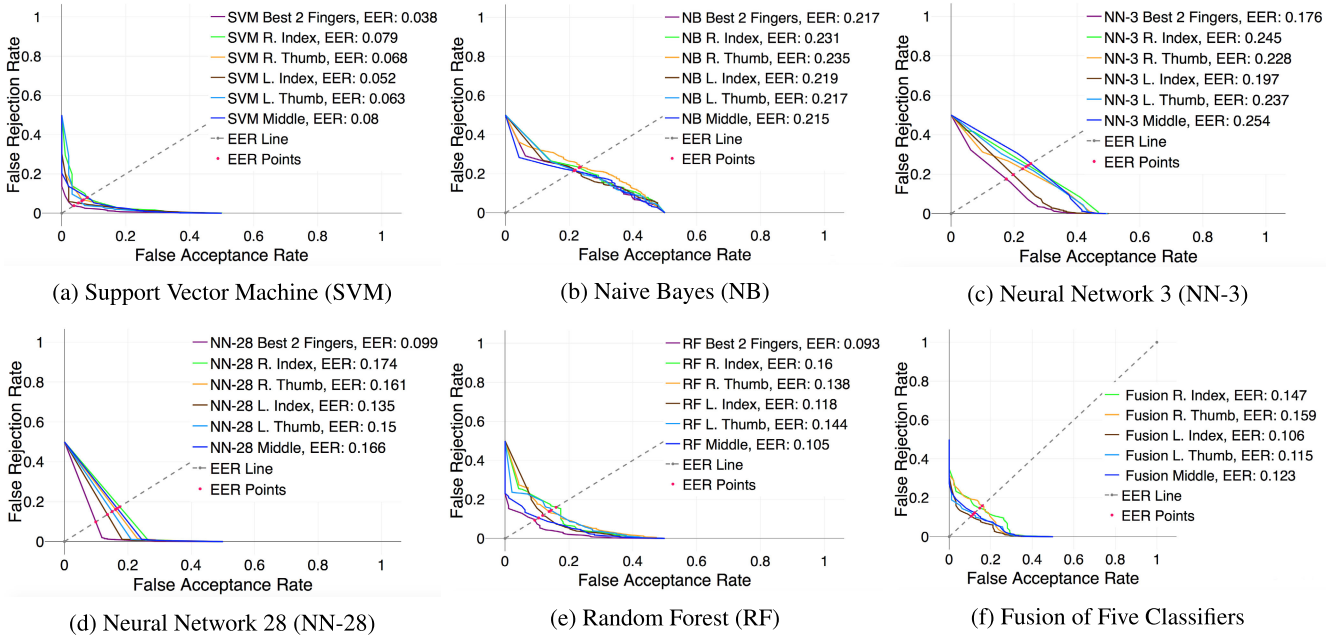


FIGURE 2. DET curves obtained by five machine learning classifiers, where the experiment was performed on Dataset 1. Figures (a)-(e) include six DET curves each: five curves generated from five fingers and one curve generated from the fusion of best two fingers. For each classifier, the best fingers were selected based on EER. Figure (f) shows five DET curves corresponding to five fingers, each curve generated from the fusion of five classifiers.

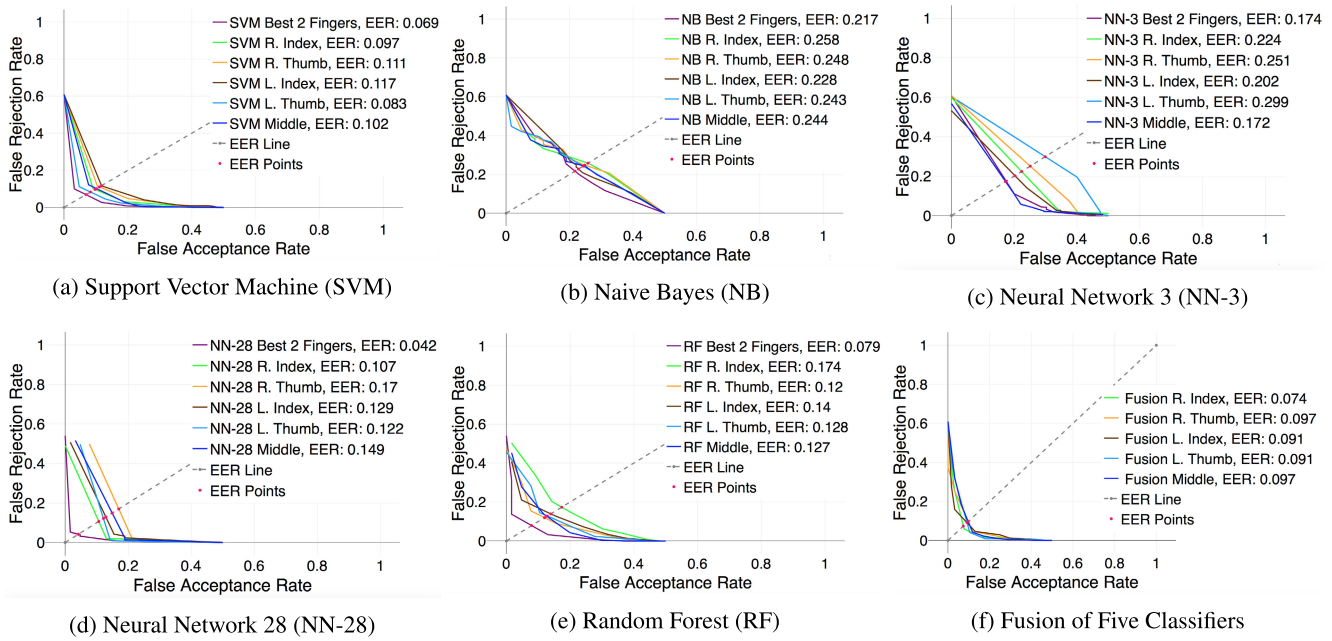
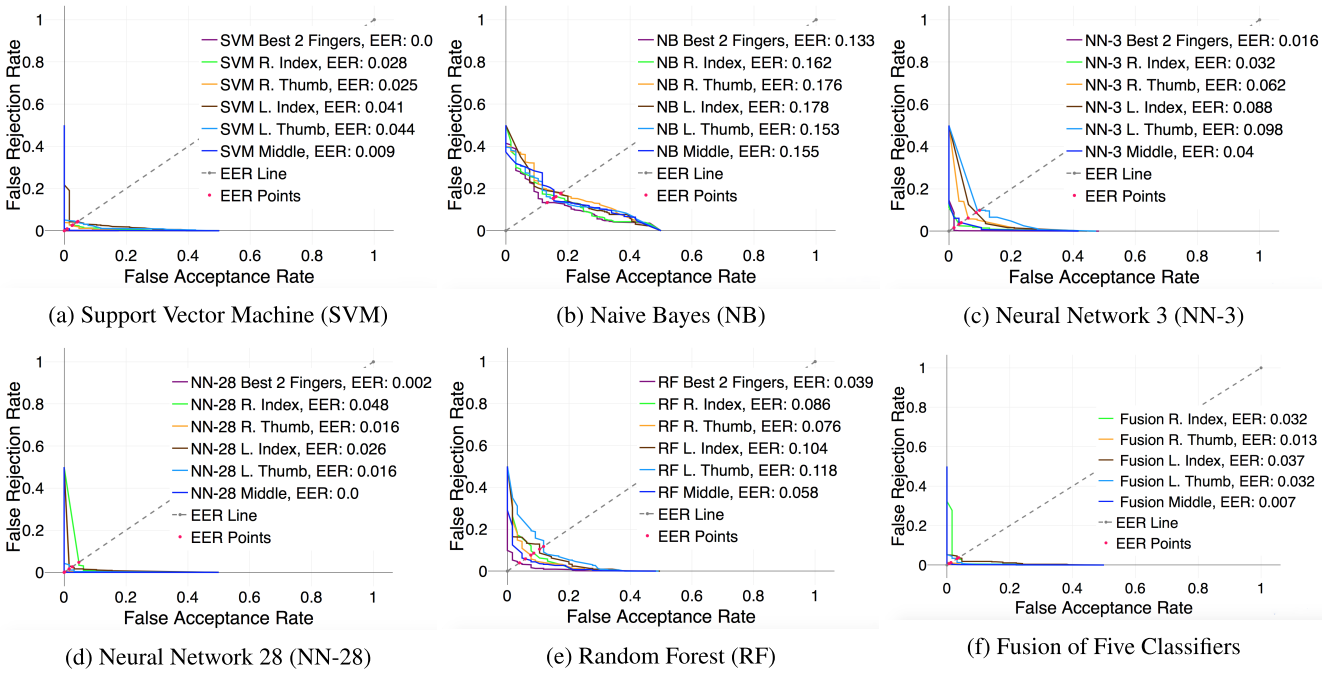


FIGURE 3. DET curves obtained by five machine learning classifiers, where the experiment was performed on Dataset 2.

time required by experiments with individual finger and the majority voting. Longer computation time was expected for fusion experiments since more data were used for training and testing than in the individual finger experiments.

VIII. DISCUSSION

Experimental results show that the Quad Swipe Pattern achieves its best performance (individually) with Neural Network 28 (NN-28) and Support Vector Machine (SVM). Generally, NN-28 takes longer training time than SVM and also it needs more training samples than SVM to be trained. In our experiment we trained both NN-28 and SVM with an equal number of training samples. Interestingly, both classifiers performed equally well. The most important thing is both classifiers achieved only 0.4% FAR which is really critical for any authentication system. At the same time,



(a) Support Vector Machine (SVM) (b) Naive Bayes (NB) (c) Neural Network 3 (NN-3)

(d) Neural Network 28 (NN-28) (e) Random Forest (RF) (f) Fusion of Five Classifiers

FIGURE 4. DET curves obtained by five machine learning classifiers, where the experiment was performed on Dataset 3.

TABLE 6. Time elapsed for experiments.

Experiment	Training (milliseconds)	Testing (milliseconds)
SVM Middle Finger	143	18
NN-3 Middle Finger	1574	4
NN-28 Middle Finger	1847	15
Naive Bayes Middle Finger	43	5
Random Forest Middle Finger	2120	364
Majority Voting Middle Finger	7192	416
SVM Best 2 Fingers	284	41
NN-3 Best 2 Fingers	2908	22
NN-28 Best 2 Fingers	2419	28
Naive Bayes Best 2 Fingers	80	19
Random Forest Best 2 Fingers	4001	675

NN-28 achieved zero FRR and SVM achieved 1.7% FRR, which is also critical because in smartphone user authentication, user convenience is one of the most important requirements.

We experimented with five fingers to see the applicability of the Quad Swipe Pattern on different fingers and different hands. While the Middle finger consistently gives the best performance, we did not find a finger/hand for which the Quad Swipe Patten works miserably. It means that the user can use any finger/hand for authentication using Quad Swipe Pattern. This is a big benefit because it gives users full flexibility of choosing any finger/hand.

Both fusion of fingers and fusion of classifiers proved to be effective on Quad Swipe Pattern. Specifically, with fusion of Right Thumb and Middle Finger, NN-28 and SVM produce zero FAR, i.e., no impostor is allowed. Fusion of classifiers using majority voting achieves the best FAR of 0.2%, which is a 50% improvement over the best individual FAR. A limitation of fusion of two best fingers is user inconvenience, where a user has to give four swipes two times (with two different fingers). Sometimes users may feel reluctant to use multiple fingers during authentication in smartphones. A limitation of fusion of classifiers is time required by multiple classifiers, which also creates user inconvenience.

A crucial part of our experimental design was to create three datasets of different sizes and evaluate how the amount of training data affects the performance of our Quad Swipe Pattern. As anticipated, we found that more training samples represent users better and improve the performance of Quad Swipe Pattern. Dataset 3, which consisted of the maximum number of training samples to create user profiles, produced a FAR of 0.4% with an individual classifier, 0.2% with a fusion of classifiers, and 0% with a fusion of the best two fingers.

A big difference between our work and existing related work is that existing work explores the plausibility of using swipe data only in continuous user authentication. In contrast, our intent in this paper is to show a path of using swipe data for point-of-entry authentication in smartphones. In addition, in devising the idea of Quad Swipe Pattern, we defined several features appropriate for this pattern and did extensive experiments to see their effectiveness.

IX. CONCLUSION

In this paper, we developed a novel point-of-entry security measure called "Quad Swipe Pattern" for smartphone users

and demonstrated its effectiveness with rigorous experiments. The results of our experiments provide very promising evidence for Quad Swipe Pattern's introduction as a new smartphone point-of-entry security measure. The most important thing to notice is that the Quad Swipe Pattern demonstrates excellent performance with numerous Machine Learning classifiers, fingers, and datasets. We have already said that the Quad Swipe Pattern has several benefits over existing point-of-entry measures for smartphone user authentication; such as (1) users do not need to memorize a password/pattern or select an easy-to-guess password/pattern, (2) shoulder surfers are no longer a threat, (3) smudge attacks become useless, (4) it is free from restrictions of lighting and face orientation, (5) it does not require extra hardware such as fingerprint scanner, (6) it is not time consuming like iris recognition, and (7) because users are habituated with swiping all day, the Quad Swipe Pattern is an easy-to use measure for them. Because of all these benefits and high performance, the Quad Swipe Pattern has the potential to be the next generation of point-of entry security measure.

ACKNOWLEDGMENT

The authors would like to thank Carl Haberfeld, a recent graduate of the Computer Science Department, Southern Connecticut State University, for his work on the data collection application that made this research possible.

REFERENCES

[1] N. Ben-Asher, N. Kirschnick, H. Sieger, J. Meyer, A. Ben-Oved, and S. Möller, "On the need for different security methods on mobile phones," in *Proc. 13th Int. Conf. Hum. Comput. Interact. With Mobile Devices Services (MobileHCI)*, 2011, pp. 465–473.

[2] A. C. Estes, *The Most Common iPhone Passcodes Are Easy to Guess*. New York, NY, USA: Atlantic, Jun. 2011. [Online]. Available: https://www.theatlantic.com/technology/archive/2011/06/most-common-ipho%ne-passcodes-are-easy-guess/351732/

[3] Z. Li, W. Han, and W. Xu, "A large-scale empirical analysis of Chinese web passwords," in *Proc. USENIX Secur. Symp. (USENIX Secur.)*, San Diego, CA, 2014, pp. 559–574.

[4] A. J. Aviv, K. Gibson, E. Mossop, M. Blaze, and J. M. Smith, "Smudge attacks on smartphone touch screens," in *Proc. 4th USENIX Conf. Offensive Technol.*, Berkeley, CA, USA: USENIX Association, 2010, pp. 1–7.

[5] A. Greenberg, *Don't Rely on an Unlock Pattern to Secure Your Android Phone*. San Francisco, CA, USA: Wired, Sep. 2017. [Online]. Available: https://www.wired.com/story/android-unlock-pattern-or-pin/

[6] C. Bonnington, *The Trouble With Apple's Touch ID Fingerprint Reader*. San Francisco, CA, USA: Wired, Dec. 2013. [Online]. Available: https://www.wired.com/2013/12/touch-id-issues-and-fixes/

[7] M. Rogers, *Why I Hacked Apple's TouchID, and Still Think it is Awesome*. Amsterdam, The Netherlands: Lookout Blog, Sep. 2013. [Online]. Available: https://blog.lookout.com/why-i-hacked-apples-touchid-and-still-think-it%-is-awesome

[8] A. Villas-Boas, *I've Been Using the iPhone X for 2 Weeks, and Face ID is a Problem*. New York, NY, USA: Business Insider, Feb. 2018. [Online]. Available: https://www.businessinsider.com/apple-iphone-x-face-id-is-a-problem-201%8-2

[9] F. Wang and X. Ren, "Empirical evaluation for finger input properties in multi-touch interaction," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst.*, Apr. 2009, pp. 1063–1072.

[10] T. Feng, J. Yang, Z. Yan, E. M. Tapia, and W. Shi, "TIPS: Context-aware implicit user identification using touch screen in uncontrolled environments," in *Proc. 15th Workshop Mobile Comput. Syst. Appl.*, Feb. 2014, pp. 9:1–9:6.

[11] M. A. Rilvan, K. I. Lacy, M. S. Hossain, and B. Wang, "User authentication and identification on smartphones by incorporating capacitive touchscreen," in *Proc. IEEE 35th Int. Perform. Comput. Commun. Conf. (IPCCC)*, Dec. 2016, pp. 1–8.

[12] A. Guo, R. Xiao, and C. Harrison, "CapAuth: Identifying and differentiating user handprints on commodity capacitive touchscreens," in *Proc. Int. Conf. Interact. Tabletops Surf. (ITS)*, 2015, pp. 59–62.

[13] L. Jain, J. V. Monaco, M. J. Coakley, and C. C. Tappert, "Passcode keystroke biometric performance on smartphone touchscreens is superior to that on hardware keyboards," *Int. J. Res. Comput. Appl. Inf. Technol.*, vol. 2, no. 4, pp. 29–33, 2014.

[14] N. Zheng, K. Bai, H. Huang, and H. Wang, "You are how you touch: User verification on smartphones via tapping behaviors," in *Proc. IEEE 22nd Int. Conf. Netw. Protocols (ICNP)*, Oct. 2014, pp. 221–232.

[15] M. J. Coakley, J. V. Monaco, and C. C. Tappert, "Keystroke biometric studies with short numeric input on smartphones," in *Proc. IEEE 8th Int. Conf. Biometrics Theory, Appl. Syst. (BTAS)*, Sep. 2016, pp. 1–6.

[16] T.-Y. Chang, C.-J. Tsai, W.-J. Tsai, C.-C. Peng, and H.-S. Wu, "A changeable personal identification number-based keystroke dynamics authentication system on smart phones," *Secur. Commun. Netw.*, vol. 9, no. 15, pp. 2674–2685, Oct. 2016.

[17] D. Buschek, A. De Luca, and F. Alt, "Improving accuracy, applicability and usability of keystroke biometrics on mobile touchscreen devices," in *Proc. 33rd Annu. ACM Conf. Hum. Factors Comput. Syst.*, Apr. 2015, pp. 1393–1402.

[18] D. Buschek, A. De Luca, and F. Alt, "Evaluating the influence of targets and hand postures on touch-based behavioural biometrics," in *Proc. CHI Conf. Hum. Factors Comput. Syst.*, May 2016, pp. 1349–1361.

[19] Y. Ku, L. H. Park, S. Shin, and T. Kwon, "Draw it as shown: Behavioral pattern lock for mobile user authentication," *IEEE Access*, vol. 7, pp. 69363–69378, 2019.

[20] C. Haberfeld, M. S. Hossain, and L. Lancor, "Open code biometric tap pad for smartphones," *J. Inf. Secur. Appl.*, vol. 57, Mar. 2021, Art. no. 102688. [Online]. Available: https://www.sciencedirect.com/science/article

[21] L. Wang, M. S. Hossain, J. Pulfrey, and L. Lancor, "The effectiveness of zoom touchscreen gestures for authentication and identification and its changes over time," *Comput. Secur.*, vol. 111, Dec. 2021, Art. no. 102462. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404821002868

[22] A. De Luca, A. Hang, F. Brudy, C. Lindner, and H. Hussmann, "Touch me once and I know it's you!: Implicit authentication based on touch screen patterns," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst.*, May 2012, pp. 987–996.

[23] M. Müller, *Information Retrieval for Music and Motion*. Berlin, Germany: Springer, 2007.

[24] N. Sae-Bae, K. Ahmed, K. Isbister, and N. Memon, "Biometric-rich gestures: A novel approach to authentication on multi-touch devices," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst.*, May 2012, pp. 977–986.

[25] 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 Security*, vol. 8, no. 1, pp. 136–148, Jan. 2013.

[26] M. Shahzad, A. X. Liu, and A. Samuel, "Secure unlocking of mobile touch screen devices by simple gestures: You can see it but you can not do it," in *Proc. 19th Annu. Int. Conf. Mobile Comput. Netw. (MobiCom)*, 2013, pp. 39–50.

[27] M. Antal and L. Z. Szabó, "Biometric authentication based on touchscreen swipe patterns," *Proc. Technol.*, vol. 22, pp. 862–869, Jan. 2016.

[28] R. Kumar, V. V. Phoha, and A. Serwadda, "Continuous authentication of smartphone users by fusing typing, swiping, and phone movement patterns," in *Proc. IEEE 8th Int. Conf. Biometrics Theory, Appl. Syst. (BTAS)*, Sep. 2016, pp. 1–8.

[29] P. Siirtola, J. Komulainen, and V. Kellokumpu, "Effect of context in swipe gesture-based continuous authentication on smartphones," 2019, *arXiv:1905.11780*.

[30] M. A. Rilvan, J. Chao, and M. S. Hossain, "Capacitive swipe gesture based smartphone user authentication and identification," in *Proc. IEEE Conf. Cognit. Comput. Aspects Situation Manage. (CogSIMA)*, Aug. 2020, pp. 1–8.

[31] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proc. 5th Annu. workshop Comput. Learn. Theory (COLT)*, 1992, pp. 144–152.

[32] H. Zhang, "The optimality of naive Bayes," in *Proc. 17th Int. Florida Artif. Intell. Res. Soc. Conf., (FLAIRS)*, vol. 2, Jan. 2004, pp. 1–6.

[33] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Oct. 2011.
[34] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "API design for machine learning software: Experiences from the scikit-learn project," in *Proc. ECML PKDD Workshop, Lang. Data Mining Mach. Learn.*, 2013, pp. 108–122.
[35] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.
[36] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognit. Lett.*, vol. 27, no. 8, pp. 861–874, Jun. 2006.
[37] A. Martin, G. Doddington, T. Kamm, M. Ordowski, and M. Przybocki, *The DET Curve in Assessment of Detection Task Performance*. Fort Belvoir, VA, USA: Defense Technical Information Center, 1997.

JAMES F. BROWN received the B.S. degree in computer science from Southern Connecticut State University, New Haven, CT, USA, in May 2018, and the M.S. degree in computer science from the Dartmouth College, Hanover, NH, USA, in 2020. His research interests include smartphone user authentication, behavioral biometrics, machine learning, and computer vision and unmanned flight.

MD S. HOSSAIN (Senior Member, IEEE) received the M.S. degree in computer science and the Ph.D. degree in computational analysis and modeling from Louisiana Tech University, Ruston, LA, USA, in 2012 and 2014, respectively. He is currently an Associate Professor in computer science with Southern Connecticut State University, New Haven, CT, USA. His research interests include multi-biometric verification, behavioral biometrics, user authentication in smartphones, machine learning, and computer vision.

LISA LANCOR (Member, IEEE) received the M.S. and Ph.D. degrees in computer science and engineering from the University of Connecticut, Storrs, Connecticut. She is currently a Professor and the Chairperson of the Computer Science Department, Southern Connecticut State University (SCSU), New Haven, CT, USA. Her research interests include penetration testing and ethical hacking, user authentication in smartphones, and the scholarship of teaching and learning. She is a member of SIGCSE.