GANTouch: An Attack-Resilient Framework for Touch-Based Continuous Authentication System

Mohit Agrawal, Pragyan Mehrotra, Rajesh Kumar, *Member, IEEE*, and Rajiv Ratn Shah, *Member, IEEE*

Abstract—Previous studies have shown that commonly studied (vanilla) implementations of touch-based continuous authentication systems (V-TCAS) are susceptible to active adversarial attempts. This study presents a novel Generative Adversarial Network assisted TCAS (G-TCAS) framework and compares it to the V-TCAS under three active adversarial environments viz. Zero-effort, Population, and Random-vector. The Zero-effort environment was implemented in two variations viz. Zero-effort (same-dataset) and Zero-effort (cross-dataset). The first involved a Zero-effort attack from the same dataset, while the second used three different datasets. G-TCAS showed more resilience than V-TCAS under the Population and Random-vector, the more damaging adversarial scenarios than the Zero-effort. On average, the increase in the false accept rates (FARs) for V-TCAS was much higher (27.5% and 21.5%) than for G-TCAS (14% and 12.5%) for Population and Random-vector attacks, respectively. Moreover, we performed a fairness analysis of TCAS for different genders and found TCAS to be fair across genders. The findings suggest that we should evaluate TCAS under active adversarial environments and affirm the usefulness of GANs in the TCAS pipeline.

Index Terms—Continuous authentication, behavioral biometrics, touchstrokes, adversarial attacks, fairness, and GANs.

I. INTRODUCTION

USER authentication is an established area of research. Various methods have been used, including PIN, password, fingerprint, and face. With the changing landscape of human-computer interaction, the need for non-intrusive and continuous authentication systems is rising and evident. The study of behavioral footprints originated from human-computer-interaction for identity authentication has become an interesting area of research over the past decade. Among several behavioral footprints (e.g., gait [1], keystroke [2], touch-stroke [2], voice [3], body-movements [4]), touch-strokes have been widely studied and have shown promise for non-intrusive continuous authentication [2], [5], [6], [7], [8], [9], [10].

A. Continuous Authentication via TCAS

One of the significant drawbacks of traditional means (e.g., fingerprint, face, PIN, password) is that they require user attention. In other words, they are intrusive. The other significant weaknesses are that they offer only entry point authentication. One can use coercion, intoxication, social engineering, or other means to unlock the device and use it afterward. Thus, the researchers have been focusing on developing non-intrusive continuous authentication systems that are user-friendly and *resilient to active adversaries*. The non-intrusiveness here means that the authentication system would require minimal or no attention from users to verify their identity. The continuous part implies that the user's identity would be verified at frequent time intervals, either fixed or triggered by user actions on the device. Multiple continuous impostor alarms would lock users out of the system or require users to present additional credentials to continue. The frequency of the authentication and the number of impostor alarms required to lock users out of the system (say n_a) depend on the application scenario. For example, the frequency of the authentication would be relatively high in a high-security environment (e.g., military bases), and n_a would be low. In contrast, in a low-security environment (e.g., a defensive driving course), the frequency of authentication would be kept low and n_a high.

The contiuous authentication error rate need not be close to zero (i.e., comparable to fingerprint or face), which is often expected from an entry-point authentication system because an attacker will have to bypass multiple checks during a meaningful adversarial session of a continuous authentication system. The chances of the attacker doing so would be $p * q^n$ for n number of checks assuming q is the probability of bypassing a check in a continuous authentication setup, and initial verification was using the entry-point authentication system. The value of $p * q^n$ will decline exponentially and catch up or will become lower than the probability of the attacker fooling the entry-point authentication system [11], [12]. Besides, continuous authentication systems do not have to replace entry-point authentication systems. Instead, they can be deployed with entry-point authentication systems to achieve higher security and usability.

The suitability of touchstrokes for non-intrusive continuous authentication has been credited to its universality, collectability, distinctiveness, acceptability, permanence,

performance, and difficulty in reproduction by someone else [5], [13], [14], [15]. The authentication systems proposed in most previous studies consisted of a typical machine learning pipeline, i.e., data collection, feature engineering, classification, and performance evaluation. The introductory studies [5], [6], [16] focused primarily on collecting touchstrokes while users browsed pages, images, and answered questions; extracting a set of features from each touch stroke; using the feature vectors to train and test authentication models, and evaluating the models using genuine reject, i.e., False Reject Rate (FRR) and impostor accept, i.e., False Accept Rates (FAR). Later studies [2], [7], [8], [10], [13], [17] explored several variants such as usage contexts (sitting and walking) [10], separate templates for different types (left, right, up, and down) of swipes [7], fusion with phone movements [1], [2], and bench-marking different classifiers on multiple datasets [13].

The majority of the previous studies have treated identity authentication as a two-class problem [5], [7], [13]. In contrast, the rest have considered it a one-class classification problem [1]. Two-class classification-based approached achieved lower authentication error rates [1]. The performance of authentication systems was reported in terms of Equal Error Rate (EER–a point on ROC where FAR and FRR are equal) [5], [10] or [1] Half Total Error Rates (HTER–an average of false accept and false reject rates) [18]. EER is generally used for setting the threshold during training/validation as we cannot change the threshold during testing. HTER is recommended for reporting testing performance [18], [19]. To summarize, previous studies suggested that touchstrokes are a viable means for non-intrusive continuous authentication and have reported average error rates around 10% percent, which could be good enough, especially for continuous authentication in the civilian domain [20]. The problem, however, is that the studies have assumed the non-existence of active adversaries. Since the data generated or stored on smart devices are invaluable, active adversaries would likely exist. The following section describes some adversarial scenarios.

B. Adversarial Scenarios for TCAS

The plausible attacks on TCAS can be grouped into three categories based on time, expertise, and equipment that the attackers expend [2], [5], [7], [21], [22], [23], [24]. In the specific case of TCAS, an attacker needs to meet one or more of the following requirements. **R1:** ability to inject data into the authentication pipeline [22], **R2:** access to the target's biometric samples [24], and **R3:** reproduction of samples by training imitators (human, machine, or human+machine) in real-time [23]. Based on the amount of effort needed to meet these requirements, ongoing discussion on the Strength of Function for Authenticators by the National Institute of Standards and Technology (NIST) [21], and a recent survey [25], we group the adversarial scenarios into the following three groups.

1) Minimal-Effort Attack: Attackers need to meet the first requirement i.e., **R1:** ability to inject data into the authentication pipeline [2], [5], [7], [22], [24], to launch attacks falling under this category. Attack strategies that have been studied

for TCAS in the past and fall under this category are listed and described below:

Zero-effort attack: under this attack, the attack vectors are randomly borrowed from each possible impostor (users other than the genuine user). The name zero-effort means that the impostors made zero effort to copy or imitate the genuine user. Consequently, zero-effort is one of the most widely adopted and studied attacks, primarily due to convenience. Consequently, most previous studies have evaluated and reported TCAS's performance under zero-effort attack [2], [5], [7]. Therefore, the performance under a zero-effort attack is considered the baseline performance of TCAS in this paper.

Population-based attack: in this attack, the attack vectors are generated by taking feature vectors from all possible impostors into account. For example, attack vectors can be created by taking means of each of the individual features [24]. It is similar to creating a master key (like creating a master face by averaging all possible faces available to breach face-based authentication). Alternatively, one can create multiple groups of impostors using clustering techniques to create multiple master keys as suggested in [26] for gait-based biometrics. This type of attack assumes that the attackers have access to public datasets.

Random-vector attack: in this attack setup, the attackers generate random attack vectors by utilizing prior knowledge, i.e., the length of the feature vectors and range of feature values. Researchers often scale feature values in a fixed range, such as k-nearest neighbors demand so, as some classification algorithms. Zhao *et al.* [22] evaluated the impact of a Random-vector attack on TCAS and concluded that a Random-vector attack is highly effective on behavior-based authentication systems.

2) Moderate-Effort Attack: This category of attacks is required to meet the first two criteria, i.e., **R1:** ability to inject data into the authentication pipeline and **R2:** access to the target's biometric samples. The performance of TCAS against this category of attacks has not been reported. This attack category has been mostly studied on behavioral biometrics other than touchstrokes. We include the description of this attack category for completeness.

Snoop-forge-replay: This kind of attack has been studied in the context of keystroke-based continuous authentication systems [27]. As the name suggests, this attack works in three steps. *Snoop:* the attacker uses social engineering or other possible means to gain access to the biometric samples (or feature vectors). *Forge:* the attacker reproduces desired number of attack vectors by using the stolen genuine samples. *Replay:* the attacker then replays/feeds the forged samples to the authentication API for the time it wants to gain access to the resources protected by the continuous authentication system.

3) High-Effort Attack: This category of attacks is difficult to launch as the attackers need to meet **R2:** access to the target's biometric samples and **R3:** reproduction of samples by training imitators in real-time. These attacks are difficult to detect because they do not require any modifications to the device or leave any footprints to be traced later. These attacks involve training individuals or machines (robots) or one with the help of the other to reproduce samples that are close

enough to that of the target or an average gestures derived from publicly available databases [23], [24]. Serwadda *et al.* [24] trained a Lego robot to match the swipes of the target user. In comparison, Khan *et al.* [23] trained human imitators to produce the attack vectors. One could combine and attack TCAS using a robot imitator assisted by humans or robots.

C. Possible Countermeasures

Previous studies such as [28] have suggested that biometric systems based on raw data level distance-based matching are more resilient to the attacks; however, they exhibit very high error rates; in general, [22], [29], [30]. On the other hand, machine learning-based matches achieve much lower error rates; therefore, they are heavily used for implementing TCAS [7], [13], [31] than the distance-based matches. The primary issue with the previous machine learning-based TCAS implementation is that they assumed that Random-vector would belong to the impostor class. However, Zhao *et al.* [22] argued that Random-vectors might also belong to the genuine class. Therefore, we chose to evaluate the machine learning-based implementations of TCAS under the most common adversarial scenarios.

Two approaches have been suggested as a possible defense against minimal effort attacks on machine learning-based TCAS. The first utilizes two Generative Adversarial Networks (GANs) [32], [33] (see Figure 1). The second focuses on reducing the acceptance region by generating synthetic data (noise) around the genuine samples and considering the generated data as impostors [22]. The reduced acceptance region decreases the chances of classifying any Random-vector as genuine.

To summarize, there are multiple ways to implement and attack TCAS. This paper focuses on machine-learning-based implementations of TCAS and its evaluation under minimal-effort attack scenarios, including zero-effort, Population, and Random-vector attacks on multiple datasets under the same and cross dataset scenarios. We could launch and evaluate only minimal-effort attacks in this paper. The evaluation of the moderate or high-effort attack is a tedious task as it requires attack data collection from trained human or robot imitators. We plan to investigate this part in the future.

D. Main Contributions

The main contributions are as follows:
- We summarize traditional TCAS implementations, establish a classification of possible adversarial scenarios for TCAS, and summarize the existing defense approaches.
- We implement a traditional machine learning pipeline for TCAS. We refer to the same as vanilla TCAS (V-TCAS) hereafter and test the same under three adversarial scenarios, viz. Zero-effort, Population-based, and Random-vector. V-TCAS's false acceptance increased significantly under these attack scenarios.
- Next, we implement (extend the idea presented in the conference paper [33]) a novel Generative Adversarial Networks assisted TCAS framework (G-TCAS) and test the same under the three aforementioned adversarial scenarios. The results suggest that G-TCAS is more resilient to adversarial environments than V-TCAS.
- We benchmarked four widely studied classifiers (each with a diverse learning paradigm). The superiority of G-TCAS over V-TCAS was evident across the experimental setups.
- Additionally, we analyze the fairness of V-TCAS and G-TCAS using kernel density plots, only to find out that TCAS is fair among different genders.

The rest of the paper is structured as follows. Section II discusses the closely related works. Section III presents the design of experiments. Section IV presents and discusses the results, respectively. Finally, we conclude the paper and provide future research directions in Section V. The code is publicly available.[1]

II. RELATED WORK

This work extends the idea presented in the conference paper [33] in the following dimensions:

Additional attack scenario: The conference paper evaluated V-TCAS and G-TCAS under Zero-effort and Population-based adversarial scenarios. We extend the analysis to random-input attacks, which were shown to be very effective in penetrating V-TCAS in a recent study [22].

Additional datasets: In the conference paper [33], we had used only two datasets viz. Serwadda-touch and BBMAS-Touch. In this paper, we include two more datasets, viz. Hand Movements, Orientation, and Grasp (HMOG), and UMDAA-02Touch Datasets. The additional datasets were used to create the population-based attack environment.

Gender-wise fairness analysis and statistical significance: To the best of our knowledge, no prior study has explored whether TCAS discriminates between different genders. Therefore, we conduct additional analysis on V-TCAS and G-TCAS to determine whether TCAS is fair across gender. Unfortunately, the analysis consisted of males and females and the BBMAS-Touch dataset. Because none of the publicly available touch stroke datasets included contained gender information (to the best of our knowledge). We also include a series of kernel density plots to show that the results and conclusion hold across users and gender groups.

Besides [33] and the literature cited in it, we review papers that have investigated the vulnerability of TCAS. Additionally, we discuss studies that have utilized GANs as a countermeasure beyond TCAS.

Zhao *et al.* [22] demonstrated that previously studied designs of behavioral pattern-based authentication systems, including TCAS, are susceptible even to uniform Random-vectors. Their investigation showed that the acceptance region of the machine learning models is much bigger than the one occupied by the genuine samples. In other words, the probability of the Random-vector being accepted as a genuine is much higher than the false accept rate of the model. They demonstrated that if attackers know the length of the feature space and the range of values each feature takes, they can launch a successful attack by generating random feature vectors. The

[1]https://github.com/midas-research/GANTouch-TBIOM

TABLE I
THE LIST OF DATASETS USED IN THIS STUDY, # OF SUBJECTS, AVG.
NUMBER OF (VALID) SWIPE GESTURES PER USER, AND
DEMOGRAPHICS AVAILABILITY

Dataset	Users	Swipe/User	Demographics?
BBMAS-Touch	117	173	Yes
Serwadda	190	273	No
HMOG	100	2275	No
UMDAA-02Touch	48	4063	No

idea was evaluated on gait, touch, face, and speech. Results showed that the random attack was very successful in the case of TCAS. This paper also motivated us to evaluate our defense scheme against random attacks.

Deb and Guirguis [34] conducted a preliminary study in which they applied Auxiliary Classifier Generative Adversarial Network (AC-GAN) to achieve error rates between 2 to 11% under synthetic data attack. The study, however, included only ten users randomly selected from the Touchalytics dataset, consisting of 41 users. Nonetheless, the paper suggested that there is a promise in using GANs to train user authentication models. Gomez-Alanis *et al.* [35] concluded that GAN-based automatic speaker verification models are more robust against original and adversarial spoofing attacks. Last but not least, GAN-based defense has been applied in the image classification area to enhance the robustness of classification models against black-box and white-box adversarial attacks [36].

III. DESIGN OF EXPERIMENTS

Most of the previously proposed implementations of TCAS consist of a typical machine learning pipeline, including data collection and preprocessing, feature engineering, classification, and performance evaluation. The details of these steps are as follows.

A. Datasets

This study primarily uses BBMAS-Touch [37]. The decision to use this dataset as the main dataset was based on several factors, such as it has a good number of users (117), the number of swipes per user (173), and gender information which was desired for gender/fairness analysis, and the data were collected in a realistic setup. Three more datasets, viz. Serwadda [7], HMOG [10], and UMDAA-02Touch [17] were used to create cross dataset zero-effort adversarial environment. A summary of these datasets is provided in Table I besides briefly describing each dataset in the following paragraphs.

BBMAS-Touch [37]: This dataset consists of the touch portion of Syracuse University and Assured Information Security-Behavioral Biometrics Multi-Device, and Multi-Activity Data (SU-AIS BB-MAS) [37]. The participants were handed over a phone loaded with the data collection app during the data collection. The participants typed two fixed sentences.

Then the participants presented a series of ten questions that required varying cognitive loads to be answered with a minimum of 50 characters. The exercise required the participants to swipe between questions. The data collection app implicitly recorded touch, keystroke, and corresponding movements (accelerometer and gyroscope readings) throughout the process.

Serwadda [7]: This dataset contains touch gestures collected from 190 participants. The participants used *Google Nexus S*, an Android-based smartphone, to answer a series of multiple-choice questions after reading or scrolling through images and textual paragraphs. Browsing through the passages and images to answer the questions generated hundreds of touch gestures. The data collection exercise consisted of two independent sessions, separated by at least a day. Each participant generated 400 swipes on average.

HMOG [10]: This dataset was created using 10 *Samsung Galaxy S4*, Android-based phones. A total of 90 individuals participated who were randomly assigned a reading, writing, or map navigation session. Each session lasted about 5-15 minutes. The participants were either sitting or walking while working on each session. Every participant performed 24 sessions (eight for each reading, writing, and map navigation session). The recorded signals consisted of raw touch events, tap gestures, scale gestures, scroll gestures, fling gestures, keypresses, and corresponding device movements captured by inertial sensors, viz. accelerometer, gyroscope, and magnetometer.

UMDAA-02Touch [17]: This dataset consists of swipe gestures collected from 48 participants for two months. The participants were not given any particular task to generate swipes. This dataset, thus, consists of touch gestures closer to how users interact with the phone through touch. Google Nexus 5, an Android-based phone, was used in the data collection. The data was collected for over two months, unrestricted.

B. Separation of Training and Testing Data

BBMAS-Touch, the base dataset, was divided into two parts with a 60:40 ratio, with 60% being the training dataset while 40% for testing. The training dataset was used for training the model and deciding upon the values of hyperparameters using a 5-fold cross-validation technique. The test data was kept unseen during the training phase. The testing environment used Serwadda, HMOG, and UMDAA-02Touch datasets.

C. Preprocessing and Feature Engineering

We excluded the swipes that had five or fewer data touchpoints. This process resulted in swipes that likely consisted of unique individual behavior. For BBMAS-Touch, the preprocess step removed about 10.33% of swipes resulting in 20286 swipe gestures. Other datasets were cleaned similarly. UMDAA-02Touch did not have the pressure information, so we appended zero. The next step was to extract features from the swipes to derive hidden characteristics of the swipe gestures. A swipe gesture S can be defined as a set of tuples representing n touch events between touching the screen with fingers and lifting the fingers from the screen and represented as follows:

$$S = (x, y, t, a, b)_{\mathtt{i=1\ to\ n}} \qquad (1)$$

TABLE II
THE LIST OF ALL THE FEATURES EXTRACTED FROM INDIVIDUAL SWIPES

Feature Id	Feature name	Equation	Description
1	$swipe\_duration$	t_{end} - t_{start}	Duration between start and end of a swipe
2 − 5	$start\_x, start\_y,$ end_x, end_y	$x_0, x_0, x_{n-1}, y_{n-1}$	Coordinates of a swipe
6	dp	$\sqrt{(x_{n-1} - x_0)^2 + (y_{n-1} - y_0)^2}$	Displacement of swipe
7	l	$l = \sum_{i=1}^{n-1} \sqrt{(x_{i-1} - x_i)^2 + (y_{i-1} - y_i)^2}$	Length of the swipe
8	$velocity$	$dp/(t_{end} - t_{start})$	Velocity of a swipe
9	$initial\_v$	Velocity of first 5% of the points	Initial velocity
10	$final\_v$	Velocity of final 5% of the points	Final velocity
11	$mean\_v$	$(v_x)_i = \frac{x_i - x_{i-1}}{t_i - t_{i-1}}, (v_y)_i = \frac{y_i - y_{i-1}}{t_i - t_{i-1}}$	Pairwise average velocity (magnitude)
12	$direction$	$\theta = tan^{-1}(\frac{x_{n-1} - x_0}{y_{n-1} - y_0})$	Angle of line joining start and end points
13	$area$	$A = 1/n \times \sum_{i=1}^{n} \pi \times a_i \times b_i$	Average area of the fingertip over the swipe
14	$acceleration$	$(a_x)_i = \frac{(v_x)_i - (v_x)_{i-1}}{t_i - t_{i-1}}, (a_y)_i = \frac{(v_y)_i - (v_y)_{i-1}}{t_i - t_{i-1}}$	Acceleration between start and end points
15	$mean\_a$	Pairwise average acceleration (magnitude)	Average acceleration
16	$initial\_a$	Acceleration of first 5% points	Initial acceleration
17	$final\_a$	Acceleration of final 5% points	Final acceleration
18 − 20	$aP_{25}, aP_{50}, aP_{75}$	Acceleration (aP_m) of m% swipe	Acceleration percentile
21 − 23	$vP_{25}, vP_{50}, vP_{75}$	Velocity (vP_m) of m% swipe	Velocity percentile
24	$speed$	$l/(t_{end} - t_{start})$	Speed of a swipe
25 − 26	$initial\_s, final\_s$	Speed of first 5% points, Speed of final 5% points	Initial speed, Final speed
27 − 29	$sP_{25}, sP_{50}, sP_{75}$	Speed (sP_m) of m% swipe	Speed percentile
30 − 34	$mean\_v_x, mean\_v_y,$ $mean\_a_x, mean\_a_y,$ $mean\_d$	Average of v_x, v_y, a_x, a_y, dp	Mean of features
35	max_d	$d_i = \frac{|y_i - m \times x_i - c|}{\sqrt{1 + m^2}}$	Maximum of deviations
36 − 38	$v_x P_{25}, v_x P_{50}, v_x P_{75}$	Velocity ($v_x P_m$) of m% swipe	Mean velocity percentile
39 − 41	$v_y P_{25}, v_y P_{50}, v_y P_{75}$	Veocity ($v_y P_m$) of m% swipe	Mean velocity percentile
42 − 44	$a_x P_{25}, a_x P_{50}, a_x P_{75}$	Acceleration ($a_x P_m$) of m% swipe	Mean acceleration percentile
45 − 47	$a_y P_{25}, a_y P_{50}, a_y P_{75}$	Acceleration ($a_y P_m$) of m% swipe	Mean acceleration percentile

where $x, y, t, a,$ and b represent, x-coordinate, y-coordinate, time, and major-axis and minor-axis of the fingertip of each touch event, respectively.

Building upon previous studies [5], [7], we extracted 30 features and added 17 new features. The process resulted in 47 features as listed and described in Table II.

D. Class Imbalance

Since we used the rest of the users as impostors, the number of genuine feature vectors turned out to be far less than the number of impostor feature vectors. To address such as class imbalance, we used the Adaptive Synthetic Over-sampling approach for imbalanced learning (ADASYN) [38] after trying several variants of Synthetic Minority Oversampling Technique (SMOTE) [39].
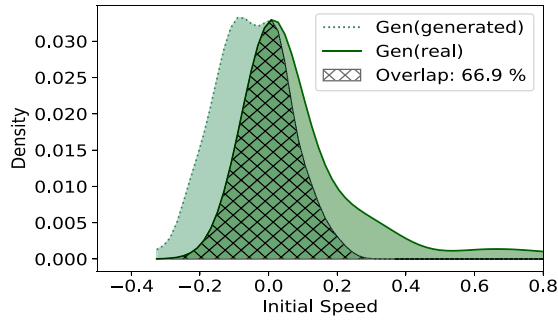
E. Choice of Classifiers

Previous studies have used several classifiers. For example, Serwadda *et al.* [7] evaluated ten classifiers. Frank *et al.* [5] used SVM and k nearest neighbors, while Fierrez *et al.* [13] used SVM, GMM, and their fusion. Kumar *et al.* [1], [2] experimented with eight classifiers. Previous studies did not agree on which classifier was the best, so we developed three criteria to select classifiers for evaluation. First, the classifiers should have achieved less than 10% error rates in the previous studies. Second, the classifier can be trained with small training data. The third criterion was the diversity of learning paradigms. The selection process resulted in

Support Vector Machine (SVM), Random Forest (RF), and Multilayer Perceptron (MLP). We added Extreme Gradient Boosting (XGB) to the list primarily because it was not tested in the previous TCAS studies and has performed very well in online data science competitions. We used these many classifiers primarily because we wanted to ensure that the idea of including GAN in the training process is not limited to certain algorithms or learning paradigms.

F. The Continuous Framework

The continuous part of the authentication systems was implemented using the sliding window scheme. A window initially contains p consecutive swipes. The following windows are created by sliding the window that drops q least recent swipes and adds q following swipes. Preliminary experimentation led us to set p and q to 5 and 1, respectively. Instead of taking an average of the features as done in the past, we concatenated them, which resulted in 235 features. These many features prompted us to use a mutual information-based feature selector in the pipeline. The number of features used for each user authentication model thus varied [29]. The varying number of features across the authentication offers another challenge for the attackers to figure out the length of the feature vectors in some adversarial environments [22]. In our implementations of those scenarios, we assumed that the attacker would have access to the feature vector length information.

Fig. 1. The system architecture of TCAS implemented in this paper. The novel part in the pipeline is highlighted with dotted blue lines, which makes use of two generative networks, viz., Genuine-GAN and Impostor-GAN. Additionally, we test the V-TCAS and G-TCAS under three adversarial environments viz., Zero-effort (traditional), Random-vector, and Population-based.

G. Training of V-TCAS and G-TCAS

The training of G-TCAS-based user authentication models is depicted in Figure 1. In comparison, the training of V-TCAS-based authentication systems excludes the GAN-based component surrounded by blue dashed lines in the training pipeline. To train the authentication model u_i, we labeled the feature vectors extracted from the training session data of u_i as genuine and the feature vectors extracted from the rest of the users, i.e., $U \setminus u_i$ impostors, where U is the set of all the users.

As rendered in Figure 1, the G-TCAS framework uses a pair of Generative Adversarial Networks (GANs). The first viz. Genuine-GAN generates swipes similar (closer) to the Genuine swipes, while the second viz. Impostor-generated swipes are similar (closer) to the impostor swipes. Genuine-GAN consist of a discriminator $D(x_l)$ and a generator $G(z_l)$. Where x_l represents feature vectors extracted from real swipes belonging to the genuine user. While z_l represents the input noise for the generator. Both $D(x_l)$ and $G(z_l)$ are trained simultaneously as they play a min-max game with the value function given in Equation (2).

$$\min_G \max_D V_l(D, G) = E_1 + E_2 \tag{2}$$

$$E_1 = E_{x \sim p_l(x)} \big[\log D(x_l) \big] \tag{3}$$

$$E_2 = E_{z_l \sim p_{z_l}(z_l)} \big[\log(1 - D(G(z_l))) \big] \tag{4}$$

where, p_l represents distribution of the generator over genuine swipes x_l, and p_{z_l} represent noise for generating (synthetic) genuine swipes.

Similarly, Impostor-GAN consists of $G(z_a)$ and a discriminator $D(x_a)$. Where, x_a represents feature vectors extracted from real swipes belonging to the impostors. While z_a represents the input noise for the generator $G(z_a)$. Both $D(x_a)$ and $G(z_a)$ are trained simultaneously as they play a min-max game

with the value function given in Equation (5).

$$\min_G \max_D V_a(D, G) = E_3 + E_4 \tag{5}$$

$$E_3 = E_{x \sim p_a(x)} \big[\log D(x_a) \big] \tag{6}$$

$$E_4 = E_{z_a \sim p_{z_a}(z_a)} \big[\log(1 - D(G(z_a))) \big] \tag{7}$$

where, let p_a represents distribution of the generator over impostor swipes data x_a and p_{za} represent noise for generating (synthetic) impostor swipes.

It is difficult to evaluate the quality of generated data by GANs, especially when we deal with non-visual data. Nevertheless, we examined the data generated by both Genuine-GAN and Impostor-GAN at the feature level using Kernel Density Estimation (KDE) plots. Figure 2 illustrates the distribution of real and generated feature values for one of the top features.

The outputs of both Genuine-GAN and Impostor-GAN are appended to genuine and impostors, respectively. Preliminary experiments suggested that the number of synthetic swipes included in the TCAS training/validation impacted the overall error rate, i.e., HTER. Therefore, we decided to consider the number of synthetic swipes as a hyperparameter during the model training and validation. The number of swipes that achieved the minimum HTER was selected. We experimented with (generated) swipes ranging between [100, 1000] and found 250 achieving the lowest validation HTER.

H. Testing of V-TCAS and G-TCAS

Once trained, each of the authentication models were tested for genuine fail and impostor pass. Test for genuine failure is straightforward. We test the models using the data (preferably collected in a different exercise than the training data) collected from genuine individuals. The percentage of failed genuine attempts is the Genuine or False Reject Rate (FRR).

(a) Genuine (real vs. generated densities).

(b) Impostor (real vs. generated densities).

Fig. 2. Visual evaluation of GAN generated data at the feature level. The illustration is only for demonstration purposes, and such overlap might not hold for all the features and users.

Test for the impostor pass, on the other hand, could be done in several possible ways. We test each model for impostor pass under three adversarial environments, viz. Zero-effort, Population, and Random-vector (see Figure 1). The percentage of successful impostor attempts is called Impostor or False Accept Rates (FAR). We compute the FAR under each adversarial environment separately. The implementation of each adversarial environment is described below.

1) Zero-Effort Attack: This is the most widely adopted adversarial environment for TCAS. As the name suggests, the authentication models are tested against a dataset produced with no intention or effort to imitate or copy the genuine users. The traditional way to implement the zero-effort adversarial environment is to consider all users except the genuine user as impostors. Previous studies used users from the same dataset as impostors, so we refer to that scenario as the same dataset zero-effort environment. An impostor can come from anywhere, i.e., any dataset. So we implemented same-dataset as well as cross-dataset zero-effort attack scenarios. The step-by-step process is described in Algorithm 1.

2) Population Attack: This adversarial environment was created in two steps. First, we computed the mean (μ_i) and the standard deviation (σ_i) for each feature (i) across all feature vectors from all the datasets except the dataset used in the training. Second, we generated feature vectors using the formula $\mu_i + r \times \sigma_i$ for each feature, where $r \in \mathcal{N}(0, 3)$. We followed this process to generate 10000 feature vectors to attack each authentication model. Algorithm 2 explains the process in more detail.

Algorithm 1 Zero_Effort_Attack(M[], D[], X[])

Input: A: list of authentication models for user $u_i \in U$
  D: List of datasets containing feature vectors
Output: C: List of dictionaries with keys (authentication models) and values (predicted labels)
$C \leftarrow []$
for *dataset in D* **do**
  temp $\leftarrow$ {}
  X $\leftarrow$ get_feature_matrix(u_i, dataset)
  X' $\leftarrow$ normalize(X)
  for *model in A* **do**
    pred_labels $\leftarrow$ []
    for *feat_vector in X'* **do**
      pred_labels.append(model.predict(feat_vector))
    end
    temp[model] $\leftarrow$ pred_labels
  end
  C.append(temp)
end
return C

Algorithm 2 Population_Attack(A[], M, S, N)

Input: A[]: list of authentication models for user $u_i \in U$
  M: list of means, computed over all the datasets
  S: list of standard dev corresponding to the means
  N: number of feature vectors to be generated
Output: C: A dictionary with keys (authentication models) and values (predicted labels)
$C \leftarrow \{\}, X' \leftarrow []$
for $i \leftarrow 0$ *to N* **do**
  attack_vector $\leftarrow$ []
  for μ, σ *in zip(M,S)* **do**
    $r \leftarrow \mathcal{N}(0, 3)$
    attack_vector.append($\mu[j] + r \times \sigma[j]$)
  end
  X'.append(attack_vector)
end
X' $\leftarrow$ normalize(X')
for *model in A* **do**
  pred_labels $\leftarrow$ []
  for *feat_vector in X'* **do**
    pred_labels.append(model.predict(feat_vector))
  end
  C[model] $\leftarrow$ pred_labels
end
return C

---

3) Random-Vector Attack: This adversarial environment was motivated from [22]. We selected uniform random values between 0 and 1 for each feature in the feature vector to implement this. We followed this process to generate 10000 feature vectors to attack each authentication model. Algorithm 3 shows the steps to implement the random-vector attack.

I. Performance Evaluation

To evaluate the performance of the authentication systems, we used FRR (defined in Equation (9)) and FAR (defined in Equation (8)), respectively. We also report Half Total Error Rate (HTER), defined in Equation (10), recommended by Bengio *et al.* [18] so we can compare different implementations and attack environments. It is worth noting that under attack environments, the FRRs remain unaffected. Therefore, we report the FRR separately in addition to reporting the FARs

Algorithm 3 Random_Vector_Attack(A[], L, N)

Input: A[]: list of authentication models for user $u_i \in U$
    L: length of feature vector to be generated
    N: number of feature vectors to be generated
Output: C: A dictionary with keys (authentication models) and
  values (predicted labels)
C ← { }, X' ← []
for $i \leftarrow 0$ *to* N **do**
    attack_vector ← []
    for $j \leftarrow 0$ *to* L **do**
        $r \leftarrow \mathcal{U}(0, 1)$
        attack_vector.append(r)
    end
    X'.append(attack_vector)
end
for *model in A* **do**
    pred_labels ← []
    for *feat_vector in X'* **do**
        pred_labels.append(model.predict(feat_vector))
    end
    C[model] ← pred_labels
end
return C

Fig. 3. FRRs of V-TCAS and G-TCAS for different classifiers.

and HTERs for each of the adversarial scenarios.

$$FAR = \frac{number\ of\ successful\ impostor\ attempts}{total\ number\ of\ impostor\ attempts} \quad (8)$$

$$FRR = \frac{number\ of\ failed\ genuine\ attempts}{total\ number\ of\ genuine\ attempts} \quad (9)$$

$$HTER = (FAR + FRR)/2 \quad (10)$$

IV. RESULTS AND DISCUSSION

The performance of the TCAS under different adversarial environments is presented in terms of FRR, FAR, and HTER. The FRRs for each of the authentication models remain unaffected across the adversarial scenarios and turned out to be between $2.9 - 4.5\%$ for different classifiers and architectures (V-TCAS and G-TCAS) (see Figure 3). FARs and HTERs are presented in Figures 4 and 5 across different experimental setups. We break down and present our results for V-TCAS and G-TCAS based on adversarial setups. First, we present the performance of V-TCAS and G-TCAS under Zero-effort (same dataset), Population, and Random-vector attack setups. Further, we present the performance of Zero-effort for same- and cross-dataset attacks. In the end, we present the gender-level analysis of the performance of V-TCAS and G-TCAS.

A. Zero-Effort vs. Population vs. Random-Vector

Figure 4 summarizes the results obtained under each of the attack scenarios for all classifiers and architectures (V-TCAS and G-TCAS). The TCAS models achieved between $5 - 7\%$ HTERs and $7 - 12\%$ FARs for Zero-effort attacks. Population and Random-vector attack severely impacted most V-TCAS models as the FARs increased to $24 - 34\%$ and $9 - 27\%$, respectively. In contrast, the G-TCAS models show more resilience than V-TCAS across adversarial scenarios obtaining FARs between $9 - 20\%$ and $4 - 23\%$ for Population- and Random-vector attacks, respectively. The FAR heatmap (Figure 4(a)) suggests that the SVM-G-TCAS is the most robust TCAS architecture, closely followed by RF-G-TCAS, on average. Interestingly, XGB-based models did exceptionally well under the traditional (i.e., the Zero-effort setup) but did not show much resilience under the Population and Random-vector attack scenarios.

Although the Zero-effort attack serves as an appropriate baseline for comparison with literature [5], [7] one should not judge the quality of authentication models only based on the performance under the Zero-effort attack scenario. Another inference we can draw from Figure 4 is that a Population-based attack is more damaging than the Random-vector and Zero-effort (same dataset) attacks. From this set of results, we encourage future researchers to test their authentication models at least under these three different attack scenarios because the performance evaluated only under the Zero-effort attack scenario could be misleading.

B. Zero-Effort (Same) vs. Zero-Effort (Cross)

Figure 5 presents the FARs and HTERs under the same dataset Zero-effort attack and cross dataset Zero-effort attacks. As we can see, the cross dataset Zero-effort attacks cause significantly more damage than the same dataset Zero-effort attacks across the classifiers. The heatmap suggests that the traditional, i.e., same dataset, zero-effort attack setup alone is insufficient to evaluate the impostor pass rate for TCAS. Therefore, we recommend that future studies on TCAS evaluate the system under cross dataset Zero-effort attack setup and the same dataset Zero-effort attack setup.

More importantly, we can observe that G-TCAS is outstandingly resilient to the cross-dataset Zero-effort attacks compared to V-TCAS consistently across the classifiers. The heatmap clearly shows the importance of including the pair of GANs in the pipeline. In other words, the effectiveness of the proposed architecture G-TCAS is evident. The HTER heatmap suggests that SVM-based G-TCAS is the best architecture, closely followed by Random Forest-based G-TCAS. In comparison, the HTERs achieved by XGB-G-TCAS exceed 10%, followed by MLP-G-TCAS.

When multiple datasets are unavailable, the proper way to evaluate the impact of the cross-dataset zero-effort attack is by collecting more data in various experimental settings, from a different user population, or both. Alternatively, one could divide the dataset into multiple groups of users and follow a strategy similar to the k-fold cross-validation. Besides, one

Fig. 4. The performance of V-TCAS and G-TCAS under different adversarial scenarios. The benefit of including Genuine and Impostor-GANs in the pipeline, i.e., G-TCAS architecture, is evident as it consistently achieves lower FARs (in turn HTERs) across the classification algorithms and attack scenarios compared to V-TCAS (the architecture without the two GANs).
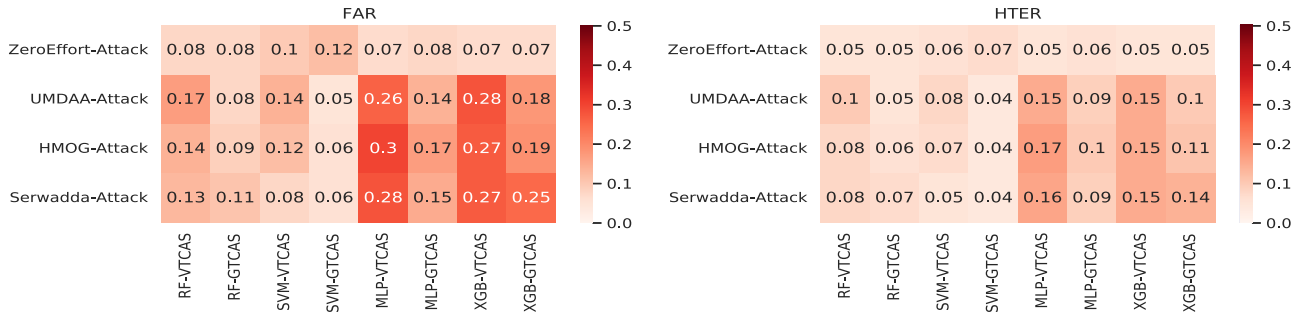

Fig. 5. The performance of V-TCAS and G-TCAS models under Zero-effort (same dataset) and Zero-effort (cross-dataset) attack scenarios. Interestingly, the Zero-effort attack using a different dataset was more damaging than the one launched using the same dataset. This finding suggests that even for Zero-effort attack scenarios, it is essential that we use different datasets because, in practice, it is very much possible.

can generate multiple datasets algorithmically using population statistics extracted from existing datasets or by altering the distribution of feature values for each user as recommended by Ballard *et al.* [40].

C. Male vs. Female

One vital aspect that has not been covered much in the TCAS literature is whether TCAS is fair among genders. One of the reasons fairness analysis has not received attention is that most of the public datasets do not contain the gender information of the participants. A recently published dataset, BBMAS-Touch, contained gender information. Therefore, we could conduct a fairness analysis. The results are presented via Figure 6. The error rates across the TCAS architectures and adversarial scenarios suggest that TCAS does not discriminate among different genders. This conclusion, however, is limited by a limited user dataset. The fairness aspect of TCAS needs to be paid attention to and studied further for people of different demographics. In the future, we would like to include a recently published dataset such as [41] which consists of demographic information of 600 users in the fairness analysis.

D. Motivation Behind the G-TCAS Architecture

In the conference paper [33], we demonstrated that the inclusion of synthetic genuine and impostor data generated by Genuine- and Impostor-GANs helped separate the data better. Consequently, G-TCAS showed more resilience than V-TCAS. Further discussion on the motivation is drawn from [22].

Technically, TCAS is trained to classify a particular region (aka acceptance region) as genuine and a separate region as an impostor (aka rejection region). Although Figure 7 presents an oversimplified scenario, it provides insight into the success of the adversarial scenarios besides the Zero-effort (same) included in this paper. As depicted in Figure 7, the Genuine-GAN helped us fill in the green stars, and Impostor-GANs helped us fill in the red stars. The classifiers, thus, were able to draw a better boundary. Recently published datasets such as [41] consisting of a significantly high number of users, gestures per user, and type of devices used in the experiment, can be used to train the generative methods better.

V. CONCLUSION AND FUTURE WORK

We evaluated V-TCAS and G-TCAS under three active adversarial environments. G-TCAS showed significantly more resilience than V-TCAS across the classifiers and adversarial environments. We also found that traditionally studied Zero-effort attack does more damage if launched using a different dataset than the same dataset the genuine user comes from. In addition, we found that TCAS is not unfair to different genders. We evaluated V-TCAS and G-TCAS under a variety of minimal effort attacks. In the future, we will assess the robustness of V-TCAS and G-TCAS under moderate and high-effort attacks. In addition, we aim to investigate whether the idea of including dual GAN in the classification pipeline extends to other behavioral biometrics such as sensor-based gait. Further, we would like to study the effectiveness of variations of GAN
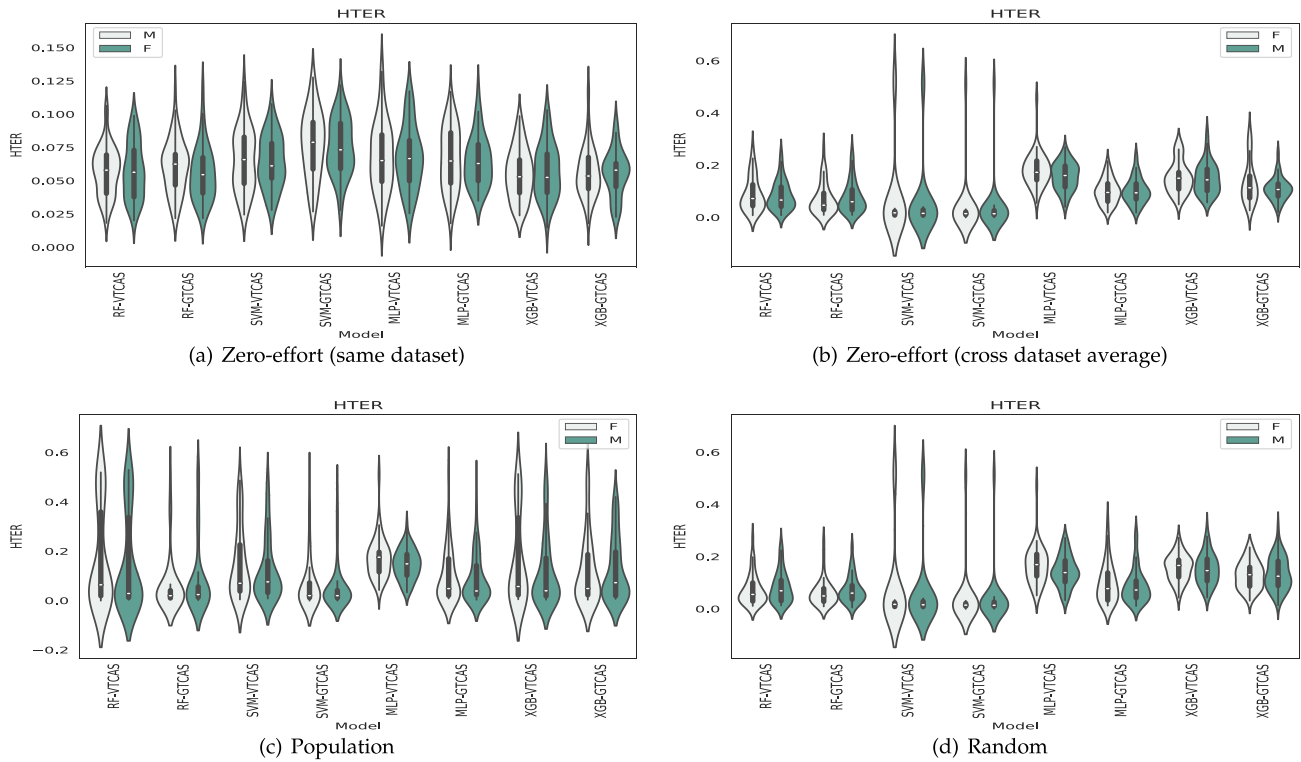
(a) Zero-effort (same dataset)

(b) Zero-effort (cross dataset average)

(c) Population

(d) Random

Fig. 6. Fairness analysis of V-TCAS and G-TCAS. Its evident from the plots that both architectures (V-TCAS and G-TCAS) regardless of the classification algorithms or adversarial environments achieve similar error rates for across genders. It worth noting that the HTERs seems breaching the boundary of zero primarily because the these plots are kernel density estimation (with Gaussian kernel) of the error rates.
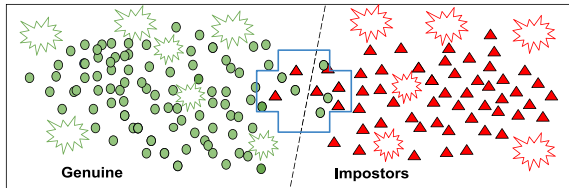
Fig. 7. An example of feature space separation by a linear boundary between two classes. An oversimplified version of TCAS demonstrating the acceptance and rejection regions and the overlap area used to compute the error rates. The GANs helped generate more cohesive genuine and impostor data points.

such as Composite Travel Generative Adversarial Networks (CT-GAN) and AC-GAN. Moreover, we plan to use explainability tools to open the pipeline and show why G-TCAS is more resilient than V-TCAS.

ACKNOWLEDGMENTS

REFERENCES

[1] R. Kumar, P. P. Kundu, and V. V. Phoha, "Continuous authentication using one-class classifiers and their fusion," in *Proc. IEEE ISBA*, 2018, pp. 1–8.

[2] R. Kumar, V. V. Phoha, and A. Serwadda, "Continuous authentication of smartphone users by fusing typing, swiping, and phone movement patterns," in *Proc. IEEE BTAS*, 2016, pp. 1–8.

[3] H. Feng, K. Fawaz, and K. G. Shin, "Continuous authentication for voice assistants," in *Proc. MobiCom*, 2017, pp. 343–355.

[4] R. Kumar, P. P. Kundu, D. Shukla, and V. V. Phoha, "Continuous user authentication via unlabeled phone movement patterns," in *Proc. IEEE IJCB*, 2017, pp. 177–184.

[5] M. Frank, R. Biedert, E. Ma, I. Martinovic, and D. Song, "Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authenticatio," *IEEE Trans. Inf. Forensics Security*, vol. 8, pp. 136–148, 2013.

[6] T. Feng *et al.*, "Continuous mobile authentication using touchscreen gestures," in *Proc. IEEE HST*, 2012, pp. 451–456.

[7] A. Serwadda, V. V. Phoha, and Z. Wang, "Which verifiers work? A benchmark evaluation of touch-based authentication algorithms," in *Proc. IEEE BTAS*, 2013, pp. 1–8.

[8] S. Mondal and P. Bours, "Swipe gesture based continuous authentication for mobile devices," in *Proc. IEEE ICB*, 2015, pp. 458–465.

[9] V. M. Patel, R. Chellappa, D. Chandra, and B. Barbello, "Continuous user authentication on mobile devices: Recent progress and remaining challenges," *IEEE Signal Process. Mag.*, vol. 33, no. 4, pp. 49–61, Jul. 2016.

[10] Z. Sitová *et al.*, "HMOG: New behavioral biometric features for continuous authentication of smartphone users," *IEEE Trans. Inf. Forensics Security*, vol. 11, pp. 877–892, 2016.

[11] R. B. Lee and W.-H. Lee, "Method and system for implicit authentication," U.S. Patent 15 428 306, Aug. 10, 2017.

[12] W.-H. Lee and R. B. Lee, "Implicit smartphone user authentication with sensors and contextual machine learning," in *Proc. 47th Annu. IEEE/IFIP-DSN*, 2017, pp. 297–308.

[13] J. Fierrez, A. Pozo, M. Martinez-Diaz, J. Galbally, and A. Morales, "Benchmarking touchscreen biometrics for mobile authentication," *IEEE Trans. Inf. Forensics Security*, vol. 13, pp. 2720–2733, 2018.

[14] A. Skalkos, I. Stylios, M. Karyda, and S. Kokolakis, "Users' privacy attitudes towards the use of behavioral biometrics continuous authentication (BBCA) technologies: A protection motivation theory approach," *J. Cybersecurity Privacy*, vol. 1, no. 4, pp. 743–766, 2021.

[15] N. Z. Gong, M. Payer, R. Moazzezi, and M. Frank, "Forgery-resistant touch-based authentication on mobile devices," in *Proc. AsiaCCS*, 2016, pp. 499–510.

[16] L. Li, X. Zhao, and G. Xue, "Unobservable re-authentication for smartphones," in *Proc. NDSS*, 2013, pp. 1–16.

[17] U. Mahbub, S. Sarkar, V. M. Patel, and R. Chellappa, "Active user authentication for smartphones: A challenge data set and benchmark results," in *Proc. IEEE BTAS*, 2016, pp. 1–8.

[18] S. Bengio, C. Marcel, S. Marcel, and J. Mariéthoz, "Confidence measures for multimodal identity verification," *Inf. Fusion*, vol. 3, no. 4, pp. 267–276, 2002.

[19] N. Poh and S. Bengio, "Database, protocols and tools for evaluating score-level fusion algorithms in biometric authentication," *Pattern Recognit.*, vol. 39, no. 2, pp. 223–233, 2006.

[20] U. Mahbub, J. Komulainen, D. Ferreira, and R. Chellappa, "Continuous authentication of smartphones based on application usage," *IEEE Trans. Biom., Behav., Ident. Sci.*, vol. 1, no. 3, pp. 165–180, Jul. 2019.

[21] "Presentation attack detection (PAD)." NIST. Accessed: Feb. 28, 2022. [Online]. Available: https://pages.nist.gov/SOFA/SOFA.html

[22] B. Z. H. Zhao, H. J. Asghar, and M. A. Kaafar, "On the resilience of biometric authentication systems against random inputs," in *Proc. NDSS*, 2020, pp. 1–18.

[23] H. Khan, U. Hengartner, and D. Vogel, "Targeted mimicry attacks on touch input based implicit authentication schemes," in *Proc. ACM MobiSys*, 2016, pp. 387–398.

[24] A. Serwadda, V. V. Phoha, Z. Wang, R. Kumar, and D. Shukla, "Toward robotic robbery on the touch screen," *ACM Trans. Inf. Syst. Security*, vol. 18, no. 4, p. 14, 2016.

[25] R. Mayrhofer and S. Sigg, "Adversary models for mobile device authentication," *ACM Comput. Survey*, vol. 54, no. 9, p. 198, 2021.

[26] T. Zhu, L. Fu, Q. Liu, Z. Lin, Y. Chen, and T. Chen, "One cycle attack: Fool sensor-based personal gait authentication with clustering," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 553–568, 2021.

[27] K. A. Rahman, K. S. Balagani, and V. V. Phoha, "Snoop-forge-replay attacks on continuous verification with keystrokes," *IEEE Trans. Inf. Forensics Security*, vol. 8, pp. 528–541, 2013.

[28] E. Pagnin, C. Dimitrakakis, A. Abidin, and A. Mitrokotsa, "On the leakage of information in biometric authentication," in *Proc. Indocrypt*, 2014, pp. 265–280.

[29] R. Kumar, C. Isik, and V. V. Phoha, "Treadmill assisted gait spoofing (TAGS): An emerging threat to wearable sensor-based gait authentication," *Digit. Threats Res. Pract.*, vol. 2, no. 3, p. 23, 2021.

[30] R. Kumar, "Treadmill assisted circumvention of wearable sensors-based gait authentication," Ph.D dissertation, Dept. Elect. Eng. Comput. Sci., Syracuse Univ., Syracuse, NY, USA, 2021.

[31] E. Ellavarason, R. Guest, F. Deravi, R. Sanchez-Riello, and B. Corsetti, "Touch-dynamics based behavioural biometrics on mobile devices—A review from a usability and performance perspective," *ACM Comput. Surveys*, vol. 53, no. 6, p. 120, 2020.

[32] I. Goodfellow et al., "Generative adversarial nets," in *Proc. NeurIPS*, 2014, pp. 1–9.

[33] M. Agrawal, P. Mehrotra, R. Kumar, and R. R. Shah, "Defending touch-based continuous authentication systems from active adversaries using generative adversarial networks," in *Proc. IEEE IJCB*, 2021, pp. 1–8.

[34] D. Deb and M. M. Guirguis, "Use of auxiliary classifier generative adversarial network in touchstroke authentication," in *Proc. ICMLA*, 2020, pp. 252–257.

[35] A. Gomez-Alanis, J. A. Gonzalez-Lopez, and A. M. Peinado, "GANBA: Generative adversarial network for biometric anti-spoofing," *Appl. Sci.*, vol. 12, no. 3, p. 1454, 2022.

[36] P. Samangouei, M. Kabkab, and R. Chellappa, "Defense-GAN: Protecting classifiers against adversarial attacks using generative models," in *Proc. ICLR*, 2018, pp. 1–17.

[37] A. K. Belman et al, "Insights from BB-MAS—A large dataset for typing, gait and swipes of the same person on desktop, tablet and phone," 2019, *arXiv:1912.02736*.

[38] H. He, Y. Bai, E. A. Garcia, and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," in *Proc. IEEE IJCNN*, 2008, pp. 1322–1328.

[39] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, no. 1, p. 321–357, 2002.

[40] L. Ballard, D. Lopresti, and F. Monrose, "Forgery quality and its implications for behavioral biometric security," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 37, no. 5, pp. 1107–1118, Oct. 2007.

[41] A. Acien, A. Morales, J. Fierrez, R. Vera-Rodriguez, and O. D. Mohatar, "BeCAPTCHA: Behavioral bot detection using touchscreen and mobile sensors benchmarked on HuMIdb," *Eng. Appl. Artif. Intell.*, vol. 98, Feb. 2021, Art. no. 104058.

Mohit Agrawal received the master's degree in computer science and engineering from the National Institute of Technology, Tiruchirappalli. He is currently pursuing the Ph.D. degree with the Indraprastha Institute of Information Technology Delhi, India. He also works as a Senior Machine Learning Engineer with Qualcomm. He was associated with India Naval Academy as a Guest Faculty Member. His research interest includes biometrics, social network analysis, computer vision, and machine learning.

Pragyan Mehrotra received the B.Tech. degree in computer science and engineering from the Indraprastha Institute of Information Technology Delhi, New Delhi, India. During his bachelor's degree, he has also worked as a Software Engineering Intern with Microsoft, Bengaluru, India. He is a Software Engineer with Amazon, Dublin, Ireland. His area of interest includes biometrics, machine learning, deep learning, cybersecurity, and image processing.

Rajesh Kumar (Member, IEEE) received the master's degree in mathematics from Louisiana Tech University, USA, and in computer applications from Jawaharlal Nehru University, India, and the Ph.D. degree in computer and information science and engineering from Syracuse University, USA. He is an Assistant Professor with Bucknell University, USA. Before joining Bucknell University, he was an Assistant Professor with Hofstra University, USA, and a Visiting Faculty with Haverford College, USA. His research focuses on harnessing the power of ever-evolving smart devices, wearables, and machine intelligence to solve security, privacy, and healthcare problems.

Rajiv Ratn Shah (Member, IEEE) received the Ph.D. degree in computer science from the National University of Singapore, Singapore. He is an Assistant Professor with the Department of Computer Science and Engineering (joint appointment with the Department of Human-centered Design), Indraprastha Institute of Information Technology Delhi, where he is the Founder of the MIDAS Lab. His research interests include multimedia content processing, natural language processing, image processing, multimodal computing, data science, social media computing, and the Internet of Things. He is the recipient of several awards, including the prestigious Heidelberg Laureate Forum and European Research Consortium for Informatics and Mathematics fellowships.