WACA: Wearable-Assisted Continuous Authentication

Abbas Acar*, Hidayet Aksu*, A. Selcuk Uluagac*, and Kemal Akkaya†
*Cyber-Physical Systems Security Lab (CSL)
†Advanced Wireless and Security Lab
Department of Electrical and Computer Engineering
Florida International University
{aacar001,haksu,suluagac,kakkaya}@fiu.edu

***Abstract*—One-time login process in conventional authentication systems does not guarantee that the identified user is the actual user throughout the session. However, it is necessary to re-verify the user identity periodically throughout a login session, which is lacking in existing one-time login systems. In this paper, we introduce a usable and reliable *Wearable-Assisted Continuous Authentication* (WACA), which relies on the *sensor-based keystroke dynamics* and the authentication data is acquired through the built-in sensors of a wearable (e.g., smartwatch) while the user is typing. The acquired data is periodically and transparently compared with the registered profile of the initially logged-in user with one-way classifiers. With this, WACA continuously ensures that the current user is the user who logged-in initially. We implemented the WACA framework and evaluated its performance on real devices with real users. The empirical evaluation of WACA reveals that WACA is feasible and its error rate is as low as 1% with 30 seconds of processing time and $2-3\%$ for 20 seconds. The computational overhead is minimal. Furthermore, WACA is capable of identifying insider threats with very high accuracy (99.2%).**

***Keywords*—continuous authentication, wearables, biometrics, keystroke dynamics, typing**

I. Introduction

An authentication mechanism, which re-verifies the user periodically without breaking the continuity of the session, is vital [1]. For example, users may share their passwords with family members, friends, colleagues [2], or an already-authenticated user may walk away without locking his/her computing platform (e.g., laptop) for a short time or may intentionally hand it to a non-authenticated co-worker trusting that s/he will not perpetrate anything nonsensical or malicious or a malicious former employee or disgruntled worker may want to use his/her former privileges. In all these cases, as long as the original login session is actively used, there is no mechanism to verify that the initial authenticated user is still the user in control of the computing terminal.

Continuous Authentication (CA)[1] is a good mechanism to re-verify a user identity periodically throughout a login session. However, none of the existing methods is deployed in real-life applications since they are either not reliable or not usable. In the literature, a number of works have been proposed for the use of biometrics in continuous user authentication [3], [4], [5]. However, one of the desired features in the continuous authentication is *non-intrusiveness* [6]. Physiological characteristics like iris pattern or fingerprint are not applicable in this manner since they can not be extracted seamlessly. More plausible works for CA would be behavioral characteristics [7], [8], [9] like typing rhythm, gait as they can be collected without interrupting the user. However, behavioral biometrics may suffer from high error rates due to their variability.

[1]CA is also sometimes called Active or Implicit Authentication in the literature.

Among all behavioral biometrics, the most promising results are proposed using keystroke dynamics [10], [11]. However, in a recent work [12], the reliability of classical keystroke dynamics is analyzed and an interface was designed to help an attacker so that the attacker can mimic the typing rhythm of a legitimate user by using the feedback provided by the interface. Indeed the usability and reliability of CA systems can be increased by exploiting off-the-shelf wearable devices. The sensors of these devices could play a key role to increase the usability in such a security context as well [13].

In this work, we introduce a *Wearable-Assisted Continuous Authentication* framework called WACA, where a wearable device (e.g., smartwatch) is used to authenticate a computer user continuously utilizing the motion sensors of the smartwatch. WACA uses *sensor-based keystroke dynamics*, where the typing rhythm of the user is captured by the motion sensors of the smartwatch worn by the user. In essence, keystroke dynamics is one of the behavioral biometrics that characterizes the users according to their typing pattern. Most conventional keystroke-based authentication schemes [14] have used *dwell-time* and *flight-time* as unique features of the users. These features are directly obtained by logging the timing between successive keystrokes. However, in WACA, the feature set is richer and more flexible since 6-axes motion sensor data can provide not only timing information, but also the key-pressing pressure, hand rotation, and hand displacement, etc. Also, instead of using the magnitude of each sensor, we used each axis data to create the different feature to increase its security against imitation attacks. Our feature set consists of 14 different sensory features from both time and frequency domains. These features are applied to 6-axes motion sensor data, obtaining 84 features in total. Finally, different distance measures are used to compare the registered and the unknown profile of the user as it was shown that they performed well in similar contexts [15], [16]. Also, in another work [17], users are classified according to the sequence of interactions (e.g., typing, scrolling), where the user wears a bracelet with motion sensors and radio. However, that work [17] has been shown as insecure in another work [18]. As explained, our work differs from other works in several ways to tackle those flaws and strengthen our design.

Contributions: The main contributions of this work are summarized as follows:

- We propose a sensor-based wearable-assisted continuous authentication framework for computing platforms, terminals (e.g., laptops, computers) with a smartwatch. We believe that this work has practical and far-reaching implications for the future of the usable authentication field.
- We propose a new variant of keystroke dynamics, called *sensor-based keystroke dynamics*. We show that *sensor-based keystroke dynamics* can be uniquely utilized

to authenticate and identify the users with extensive evaluation.

- We tested the performance and efficiency of WACA with real users. We conducted an extensive evaluation of the proposed method using a rich set of distance measuring techniques and machine learning (ML). While we used the distance measures for the authentication experiments, ML is used for the insider identification experiments.

Organization: The reminder of this paper is structured as follows: We introduce our system model in Section II. Then, the overall architecture of WACA is detailed in Section III. Section IV presents the performance, efficiency, and robustness evaluation of the WACA framework. Finally, in Section V, we conclude the paper.

II. Assumptions and Adversary Model

In this section, we explain our assumptions and the adversary model.

Assumptions: For WACA, the following assumptions are considered:

- We assume that the user wears a smartwatch, which is equipped with motion sensors and either Bluetooth or WiFi. We also assume that an app to collect the motion data is already installed on the smartwatch and it is paired with the computer that will be authenticated. For our work, we built a custom Android Wear app to collect and process the sensor data.
- We assume that by pairing devices, a secure communication channel is already established between the computer and smartwatch as well as between the computer and the remote or local authentication server. This secure communication channel should keep the sensor data secure in both transition and at rest.
- It is assumed that the system has a first authentication factor. The first factor could be one of the password-, token-, or biometric-based systems.

Adversary Model: In this paper, the primarily considered adversary model is an attacker who somehow bypassed the first factor (e.g., password, token) of the authentication system and it has a physical access to the computing terminal. The attacker is likely to be an insider or co-worker, but it can also be an outsider, just passing by the victim's computer. Attacker's goals can include, but not limited to, trying to get some important information from the victim's computer, taking action on behalf of the victim, or trying to get access to the assets that s/he does not have permission (i.e., privilege abuse). More specifically, we consider the following attack scenarios by considering WACA is deployed in a real world system:

- *Attack Scenario 1:* The victim is one of the employers and forgets to lock his computer and *an outsider* (e.g., a mail courier) who is just passing through the office tries to get access to the victim's computer. In this scenario, if the attacker is not aware of WACA, s/he will attempt to use the victim's computer. If the attacker is aware of WACA, s/he will first look for the victim's smartwatch and then try to keep the system logged in.
- *Attack Scenario 2:* We consider the attacker can also be a malicious insider and thereby the attacker also has a registered smartwatch, but its typing profile is registered together with its own username. This type of attacker tries to get access to the system's assets that s/he does not have permission (i.e., privilege abuse). In this scenario, the attacker watches its victim (e.g., supervisor) for a suitable timing that its victim leaves the computer unlocked for some time to go to lunch or to get coffee etc. (aka *lunchtime attack* [19]). The attacker can either try to bypass the system via providing data from his smartwatch or can try to use the victim's smartwatch somehow obtained (e.g., can steal it or victim can leave it behind).

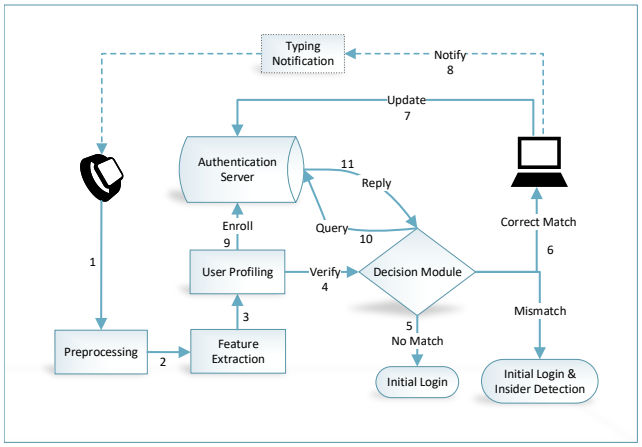

Fig. 1: WACA framework architecture and key components

III. WACA Architecture

In this section, we present the details of the WACA. WACA is a typing-based continuous authentication system using the accelerometer and gyroscope sensors of a smartwatch. WACA framework is complementary to the first factor authentication mechanisms and it is flexible to work with any first factor, including one of the password-, token-, or biometric-based systems. Note that the first factor authentication is beyond the scope of this work.

A. Overview

WACA consists of four main stages: *Preprocessing, Feature Extraction, User Profiling, and Decision Module*. These stages, which are shown in Figure 1, work as follows: First, the raw sensor data is acquired from a smartwatch (1) through an app installed on the watch. Then, the raw data is transmitted to the computer through a secure wireless channel and the rest of the stages are performed on the computer except that Authentication Server (AS) is located in a trusted place. As the collected data includes a certain level of noise, in the preprocessing stage, the raw data is cleaned up by filtering (2) Then, incoming data is used to extract a set of features (3). This set of features, namely *feature vector*, represents the characteristics of the current user profile. In the enrollment phase (9), the created feature vector is stored in the AS. In the verification phase (4), the queried user profile is dispatched from the AS to the decision module (10, 11). The decision module computes a similarity score between the returned profile and the provided profile for the current user to make a binary authentication decision (match/no match). If the decision is a no match (5), then the user's access to computing terminal will be suspended and the user will be required to re-authenticate using the primary authentication method (e.g., password). However, when the decision is a match (6) then the user's access will be maintained. The profile of the current user in the AS will be updated after the correct match

of the user profile (7). In WACA, this update frequency is a system parameter and can be set by the admin in the security policy. In this way, the user profile will be kept up-to-date over time. Whenever a typing activity is initiated on the keyboard of the computer, the smartwatch will be notified (8) again by the terminal to start over the authentication process continuously. In the following subsections, we explain the details of WACA and its key stages.

B. Data Collection

In WACA, *data collection* refers to capturing sensor readings from the user's smartwatch through a secure wireless communication channel (i.e., via WiFi or Bluetooth). An app is installed on smartwatch to listen to the physical sensors. Then, the raw sensor data is transmitted to the computer through a secure communication channel.

Each row of the collected raw data of accelerometer is represented in the format of $a\vec{c}c =< t_a, x_a, y_a, z_a >$ and gyroscope is represented as $g\vec{y}ro =< t_g, x_g, y_g, z_g >$, where t stands for timestamps and x, y, z represent the different axis values of the accelerometer and gyroscope sensors. Each of t, x, y, and z is stored as a different vector. The length of the vectors directly depends on sampling rate of the sensors and the time interval of the data collection. In WACA, the parameter *sample size* refers to the length of these vectors and it is set as a configurable parameter while the parameter *sample rate* is a constant system parameter that is characterized by the wearable device and app.

C. Preprocessing

In WACA, *preprocessing* stage refers to preparation of raw sensor readings for the next stages. It consists of cleaning and transformation of the raw data. In the cleaning part, the noise is removed. In order to remove the effect of the noise from data, we apply M-point Moving Average Filter (MAF), which is actually a simple low-pass filter and it operates by taking the average of M neighbour points and generates a single output. M-point filtering in equation form can be expressed as follows: $y[i] = \frac{1}{M}\sum_{j=0}^{M-1} x[i+j]$, where x is the raw sensor data, y is the new filtered data, and i indicates the current sample that is averaged. The filtered data becomes smoother than the raw data without altering the value at that point.

After filtering the noise, the data is transformed into appropriate forms for the next stage. Particularly, different types of sensor data are separated according to an assigned ID number during the sensor registration and then x, y, and z axes of the sensor values are recorded as different vectors e.g., $\vec{x_a} =< {x_a}^1, ..., {x_a}^n >$ and $\vec{x_g} =< {x_g}^1, ..., {x_g}^n >$ for a profile of n samples.

D. Feature Extraction & User Profiling

In WACA, *Feature Extraction* (FE) refers to the transformation of the time series raw data into a number of features. In order to create the feature vector, each feature is computed using the data vectors. As an example, the first feature is calculated from a function f, i.e., $f_1 = f(x_a, y_a, z_a, x_g, y_g, z_g)$ and the second feature is calculated from another function g, i.e., $f_2 = g(x_a, y_a, z_a, x_g, y_g, z_g)$ etc. Then, the final feature vector $\vec{f} =< f_1, f_2, ..., f_n >$ is generated using all the calculated features.

As each element of the feature vector has different ranges, some of the features can be dominant in the distance measurement. To prevent this and create a scale-invariant feature vector, we apply a normalization to the feature vector to map the interval $[x_{min}, x_{max}]$ into the unit scale [0,1]. We formulate this linear normalization process in WACA as follows: $x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$, x_{min} and x_{max} are the minimum and maximum value of the features of the user's enrolled templates.

TABLE I: Feature set extracted from sensor data in WACA.

Domain	Feature	Length
Time	Mean, Median, Variance, Average Absolute Difference of Peaks, Range, Mode, Covariance, Mean Absolute Deviation (MAD), Inter-quartile Range (IQR), correlation between axes (xy, yz, xz), Skewness, Kurtosis	12*6=72
Frequency	Entropy, Spectral energy	2*6=12
Total # of Features		**84**

After generating the final feature vector $\vec{f}$, in the user profiling stage, a user profile $\vec{p}$ is generated by adding the user ID and start and end timestamps of the data sample, i.e., $\vec{p} =< userID, t_{start}, t_{end}, \vec{f} >$. If the user is in the enrollment phase, this profile is transmitted to the AS to be stored in a database. Finally, if the user is unknown and a typing activity notification comes from the computer, the profile is passed to the Decision Module.

The feature set used in our framework is presented in Table I. These features were chosen as they performed well in similar contexts [15], [16].

E. Decision Module

The next stage in WACA is the *decision module*. The task of this stage is classifying the user as authorized or unauthorized for given credentials entered during the initial login. For the purpose of authentication, we use distance measures. The distance measure methods simply calculate the distance between two vectors or data points in a coordinate plane. It is directly related to the similarity of compared time-series data sets. The most widely used distance measure is *Euclidean Distance*. It is actually just the distance between two points in vector space and is the particular case of *Minkowski Distance*, which is expressed as follows: $distance(\vec{x}, \vec{y}) = (\sum_{i=1}^{n}(x_i - y_i)^p)^{\frac{1}{p}}$,, where $\vec{x} = (x_1, x_2, ..., x_n)$ and $\vec{y} = (y_1, y_2, ..., y_n)$ are the set of sensor observations to be compared. If $p = 2$, it is Euclidean distance and has been extensively used in the keystroke-based authentication methods. WACA calculates the distance and returns the result by comparing it with a configurable predetermined threshold value (i.e., genuine if $distance < threshold$, impostor if $distance \geq threshold$).

In addition to Euclidean and Minkowski Distances, there are several distance measurement methods utilized in biometric authentication systems which may perform differently depending on the context. Therefore, we also tested different distance metrics in our experiments to see which performs the best for WACA. Other distance metrics that we tested in our experiments are *Cosine Distance, Correlation Distance, Manhattan (Cityblock) Distance* and *Minkowski with p=5*. The performance of each one is given in Section IV-A.

IV. Performance Evaluation

We tested the performance and efficiency of WACA with twenty real users and data collected from them. We specifically evaluated WACA in terms of three metrics: (i) *How accurately can it differentiate between genuine and impostor users?* (ii) *How fast can it detect an impostor?* (iii) *How accurately can*

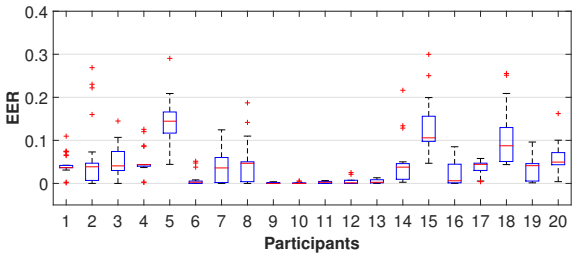

Fig. 2: EER for each participant with a sample size of 1000 using Manhattan (Cityblock) distance metric during Typing Task-1. Average EER is 0.0513.

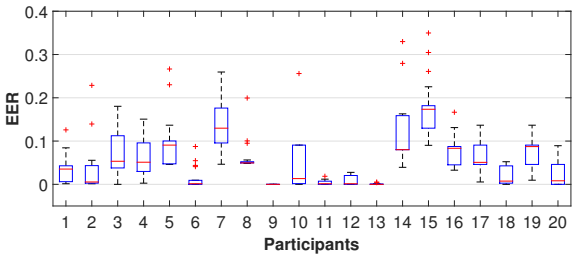

Fig. 3: EER for each participant with a sample size=1000 using Manhattan (Cityblock) distance metric during Typing Task-2. Average EER is 0.0647.

it identify an impostor? In for these purposes, we first conduct authentication experiments. In these, we measure how WACA performs when users type a different or the same text. We also analyze how the sample size and the detection technique impact WACA's performance. The effect of the sample size allowed to evaluate the quickness of WACA. Finally, we also conducted an experiment to show how successful WACA is in identifying the insider threats.

Data and Collection Methodology. In our experiments, we collected data from 20 human subjects. [2] During the collection of data, an Android Wear smartwatch with an installed data collection app was distributed to the participants and the participants were asked to type a text while the program in the smartwatch was recording its sensory data. The participants were free to choose the hand (left/right) on which they wore the smartwatch. Moreover, they were also given the freedom to adjust the sitting position and the keyboard and screen position according to their comfort levels. Throughout these experiments, we utilized a standalone qwerty keyboard to have generic results. Before typing each text, the participants were also given enough time to read the texts to make them familiar with the text as typing a familiar text is a more common activity.

The participants were involved in two typing tasks conducted in two different sessions. They were asked to type with their normal typing style without noticing that their data was recorded. The two data sets were compiled as follows:

- *Typing Task-1:* The participants were asked to type a story from a set of short and simple stories from the American Literature[3] for four minutes. The story was chosen randomly by the participants. On average, four minutes of data corresponds to 25000 samples for each participant (Total: 850000 samples).
- *Typing Task-2:* For this data set, all the participants were asked to type the same text[4] for four minutes. For each participant, almost the same amount of data is collected as Typing Task-1. This dataset is important to be able to measure the quality of the features.

Note that in all the experiments, the datasets obtained from all these tasks were always used by cross-validation techniques (i.e., partitioning the data set into iteratively chosen two sets for training and testing). Therefore, even if the same text was typed by all the participants in Typing Task-2, the compared samples always corresponded to different texts for a participant.

In our experiments, we split the collected data sets into equal size chunks, called *sample size*. It is the number of samples (i.e., row) in a chunk. Each chunk consists of 8 columns of data, two of which are timestamps and the others are 6 dimensional sensor data. Sample size is the main system design parameter in our experiments as it has a direct impact on the time required to collect data. Particularly, the time t required to collect data with the sample size can be represented as $t = sample\ size/100$ in seconds as the sampling rate in our experiments was $100Hz$.

Performance Metrics. In the authentication experiments, we used *Equal Error Rate* (EER) as it is a commonly accepted metric to assess the accuracy of WACA. EER is calculated using two metrics: False Acceptance Rate (FAR) and False Reject Rate (FRR). FAR is the rate of incorrectly accepted unauthorized users among all the illegal attempts: The increase in FAR is a direct threat to system's security level. For more valuable assets, increasing the threshold will decrease FAR. On the other hand, FRR is the rate of incorrectly rejected authorized users among all the legitimate authentication attempts. Contrary to FAR, FRR can be decreased by decreasing the value of threshold. Finally, EER is the point that gives the closest FAR and FRR point for a given threshold (ideal EER is the intersection point of FAR and FRR) and the lower the EER the better is an authentication system.

A. Results

In this section, we present and discuss the evaluation results.

Impact of the text dependency. In this experiment, our goal is to analyze how EER changes among the participants. We try to answer the question: *How does WACA perform with the typed text?* This is also a more advanced analysis of the framework and the fundamental idea than that of in Section 2.

Specifically, for this experiment, we used Typing Tasks 1 (any text) and Typing Task 2 (the same text) datasets. For each sample of a particular user, we computed the differences from other users' samples. For this purpose, we computed the NxN dissimilarity matrix, where N is the total number of samples for all the participants. The dissimilarity matrix was calculated by measuring the similarity of each sample to all the other samples using leave-one-out cross-validation method [20].

Then, for a given threshold and participant, the ratio of the rejected and accepted samples were computed to obtain FRR and FAR, respectively. This process was repeated by incrementing the threshold by 0.01 in each step for all the samples of all the participants. This gave us a set of EER for each participant. Note that in a real system, FAR/FRR rate can be tuned according to the system preferences, but here our purpose is to find an acceptable performance metric for WACA. The results are plotted in Figure 2 for Typing Task-1 and Figure 3 for Typing Task-2. Average EER for the Typing

[2]Our research study with the human subjects was conducted with the appropriate Institutional Review Board (IRB) approvals.

[3]https://americanliterature.com/100-great-short-stories

[4]https://en.wikipedia.org/wiki/The_Adventures_of_Tom_Sawyer

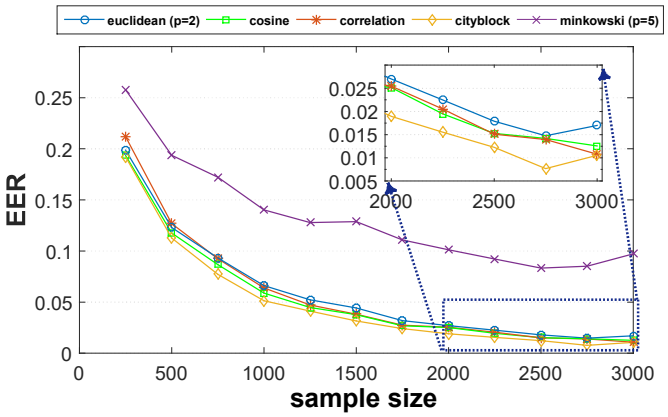

Fig. 4: Average EER according to different sample sizes using different distance metrics while users are performing Typing Task-1.

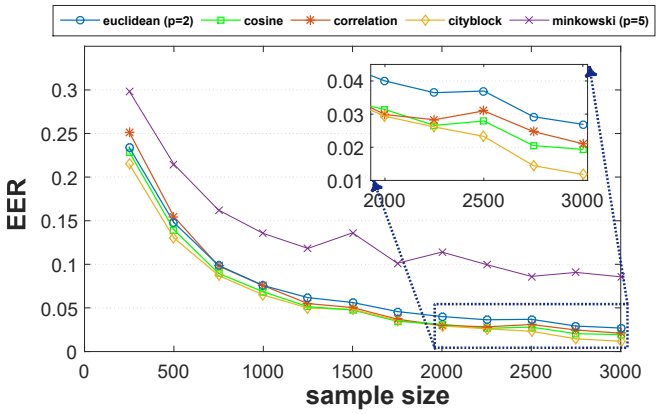

Fig. 5: Average EER according to different sample sizes using different distance metrics while users are performing Typing Task-2.

Task-1 experiment was 0.0513. Figure 3 compares the EER of participants for the Typing Task-2 experiment. Average EER for this experiment was 0.0647. Another observation from the plots is that some participants have more distinctive typing characteristics than others using both the datasets.

If we compare the ERR of each participant in both the experiments, we see that they are also close to each other, where a few of the participants perform very distinctive behaviours (e.g., participant 15). However, the overall distribution of EER over the participants is similar in both the experiments. Recall that in Typing Task-1, all the participants typed different texts, while they typed the same text in Typing Task-2.

Overall, in this analysis we report the average EERs of both the experiments are close (around %1), which supports the usability of WACA regardless of the typed text for the continuous authentication session.

Impact of the sample size and the distance measuring technique. In these experiments, *our goal was to assess how different sample sizes and the distance measuring techniques used in WACA impact the performance.* For this, we varied the sample size from 300 to 3000 and utilized five different distance measuring techniques, Euclidean (p=2), Cosine, Correlation, Cityblock, and Minkowski (p=5). Again, two types of participant datasets, Typing Task-1 (any text) and Typing Task-2 (the same text), were used. Figure 4 (Typing Task-1) and Figure 5 (Typing Task-2) present the main results when the sample size increases.

As can be seen in Figure 4 when the participants typed different texts, the EERs are generally decreasing with the increase of sample sizes as expected. The EERs go under 0.05 after the sample size of 1500 for all the distance metrics utilized except for Minkowski (p=5). Then, the EER is converging to the value of 0.01-0.02 through the sample size of 3000. In the best case, EER 0.007 is achieved with the the sample size of 2750 for the Manhattan (cityblock) distance measurement technique.

Figure 5 presents the results of the same-text experiment (Typing Task-2). As in Figure 4, the general behavior is that the EERs are decreasing with the increase of the samples. The lowest EER of 0.01 is achieved using the Cityblock distance measuring technique at 3000. We also see the convergence of EER in Figure 5 as Figure 4. Plots are starting to converge around sample sizes 1500-2000 and converging to 0.01 for Cityblock and Correlation distance measuring techniques. We also see that at 3000, 0.02 EER is obtained for Cosine and Correlation techniques. However, if shorter data collection time is of interest, a sample size of 2000, which needs 20 seconds for data collection, gives 0.03-0.04 EER. However, if we increase the sample size, both the accuracy and the data collection time are increasing. This means the time needed to catch an adversary or more generally the re-verification period would also increase. Therefore, an optimal sample size should be adjusted according to the preferences in a real application based on the usage needs or the security policies.

To conclude, the features in WACA can successfully differentiate the users from their typing rhythm with a very small error rate (1%) independent of the typed text. There is a natural trade-off between the EER and data collection time, which should be configured according to the security needs of an organization.

The accuracy of insider threat identification. As noted earlier, the insider threat detection is important in continuous authentication systems as a potential attacker is likely to be an insider. In order to effectively locate such an insider attacker within an organization where WACA is employed, an identification mechanism is needed. Hence, WACA includes *Multilayer Perceptron algorithm* (MLP) to defend against and identify insider threats. MLP is a feedforward neural network model which maps a set of input data into a set of outputs through the interconnected processing elements (neurones). We used MLP for the task of identification. Identification is a one-to-many classification task and requires a training set. We assume that the insider's data is also stored in the authentication server's database (training set) as a legitimate user. We used MLP since it gave the best results in our experiments.

In order to analyze the efficacy of WACA against insider threats, we analyzed the impact of the sample size and the size of the training data on accuracy. For this, we focused on two test scenarios that could be relevant in real investigations: *Scenario 1:* In the first scenario, we built our test model using the same text and tested again using the same text with the 5-fold cross validation technique. For this scenario, we utilized Typing Task-2 Dataset for both the training and testing. This type of scenario can be useful as all the users are asked to type a provided text and during the investigation, all users are asked again to type the same text. The results are presented in Table II. *Scenario 2:* In the second scenario, the test model was trained with the same text dataset, which is the same for all the participants and tested using random-text experiments, where each user typed a randomly chosen text. For this scenario, we utilized Typing Task-2, Typing Task-1 Datasets for training and testing, respectively. This scenario is suitable for cases where

Scenario 1: Accuracy (%)					
	Training Set				
Sample size	1	2	3	4	5
1500	77.8	**93.7**	**97.2**	**98.4**	**99.2**
1000	62.8	87.6	**93.8**	**95.3**	**97.1**
500	37.5	63.7	75.9	83.1	89.6
250	28.5	43	53.1	61.8	62.1
Scenario 2: Accuracy (%)					
	Training Set				
Sample size	1	2	3	4	5
1500	55.8	80.1	88.7	89.8	**91.8**
1000	51.7	82.7	83.2	86.1	86.8
500	29.9	51.3	66.7	73.8	76.5
250	22.1	33.6	41.9	49.8	54.1

TABLE II: The accuracy results insider threat identification experiments for different sample sizes in Scenario 1 and 2.

all the users are enrolled using the same text, but a user is verified while typing a random text. The results for this test scenario are presented in Table II.

As can be seen in Table II, in the best case, 99.2% identification rate of an insider threat can be achieved with the sample size of 1500 while the model is trained with 5 samples. Even with 2 samples of the insider, 93.7% accuracy rate can be achieved with the sample size of 1500.

Scenario 2 aims to answer the question of "*Can an insider be identified while typing a random text even if s/he is enrolled while typing a given text ?*" Table II presents the result of this question for Scenario 2. As can be seen from Table II, similar to Scenario 1, the accuracy rates increase as the sample sizes and training set increase, and the time to build model and time required to catch the attacker is also increasing. 3 training samples and the sample size is 1500 or 4 training samples with the sample size of 1000 may be the two most optimal choices for real cases.

Overall, WACA can achieve 0.01 error rate with almost 30 seconds of the data collection (see Figure 4 and 5) in the best case. If a shorter time is of interest, 0.02 error rate is achieved with 20 seconds of the data collection. Moreover, if 5 training samples with 1500 sample sizes are obtained from a potential insider threat, WACA could identify the insider with 99.2% accuracy rate while typing the provided text (see Table II) or with 91.8% accuracy rate while typing a random text (see Table II).

V. Conclusion

Wearables such as smartwatches and fitness trackers carried by individuals have grown exponentially in a short period of time. Particularly, WACA decreases the vulnerable time window of a continuous authentication system to as low as 20 seconds, prevents the privilege abuse and insider attacks and also allows the insider threat identification. Moreover, we evaluated the efficacy and robustness of WACA with real data from real experiments. The results showed that WACA could achieve 1% EER for 30 seconds or $2-3\%$ EER for 20 seconds of data collection time and error rates are as low as 1% with almost a perfect (99.2%) insider threat identification rate. Furthermore, achieved a minimal overhead on the utilization of the system's resources. As a future work, we will study privacy-aware WACA, where we will deploy privacy-preserving algorithms [21] protect user's sensitive sensor data.

VI. Acknowledgment

This work is partially supported by US National Science Foundation (NSF) under the grant numbers NSF-CNS-1718116 and NSF-CAREER-CNS-1453647. The statements made herein are solely the responsibility of the authors.

References

[1] Google, "Google eyes behavioural solution for continuous authentication," http://www.planetbiometrics.com/article-details/i/4512/, May 2016.

[2] Kaspersky, "Consumer security risks survey 2016," https://cdn.press.kaspersky.com/files/2016/10/B2C_survey_2016_report.pdf.

[3] C. M. Carrillo, "Continuous biometric authentication for authorized aircraft personnel: A proposed design," Tech. Rep., 2003.

[4] K. Niinuma and A. K. Jain, "Continuous user authentication using temporal information," in *SPIE Defense, Security, and Sensing*. International Society for Optics and Photonics, 2010, pp. 76 670L–76 670L.

[5] G. Kwang, R. H. Yap, T. Sim, and R. Ramnath, "An usability study of continuous biometrics authentication," in *International Conference on Biometrics*. Springer, 2009, pp. 828–837.

[6] D. Dasgupta, A. Roy, and A. Nag, *Advances in User Authentication*. Springer, 2017.

[7] I. C. Stylios et al., "A review of continuous authentication using behavioral biometrics," in *Proceedings of the SouthEast European Design Automation, Computer Engineering, Computer Networks and Social Media Conference*. ACM, 2016, pp. 72–79.

[8] S. Eberz, K. B. Rasmussen, V. Lenders, and I. Martinovic, "Evaluating behavioral biometrics for continuous authentication: Challenges and metrics," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*. ACM, 2017, pp. 386–399.

[9] G. Wu, J. Wang, Y. Zhang, and S. Jiang, "A continuous identity authentication scheme based on physiological and behavioral characteristics," *Sensors*, vol. 18, no. 1, p. 179, 2018.

[10] S. P. Banerjee and D. L. Woodard, "Biometric authentication and identification using keystroke dynamics: A survey," *Journal of Pattern Recognition Research*, vol. 7, no. 1, pp. 116–139, 2012.

[11] M. Rybnik, P. Panasiuk, and K. Saeed, "User authentication with keystroke dynamics using fixed text," in *Biometrics and Kansei Engineering. ICBAKE 2009.* IEEE, pp. 70–75.

[12] C. M. Tey, P. Gupta, and D. Gao, "I can be you: Questioning the use of keystroke dynamics as biometrics," *Annual Network and Distributed System Security Symposium 20th NDSS 2013, 24-27 February*, 2013.

[13] A. K. Sikder, H. Aksu, and A. S. Uluagac, "6thsense: A context-aware sensor-based attack detector for smart devices," in *26th USENIX Security Symposium (USENIX Security 17)*, 2017.

[14] P. S. Teh, A. B. J. Teoh, and S. Yue, "A survey of keystroke dynamics biometrics," *The Scientific World Journal*, vol. 2013, 2013.

[15] K. S. Killourhy and R. A. Maxion, "Comparing anomaly-detection algorithms for keystroke dynamics," in *Dependable Systems & Networks, 2009.* IEEE, pp. 125–134.

[16] A. Serwadda, V. V. Phoha, and Z. Wang, "Which verifiers work?: A benchmark evaluation of touch-based authentication algorithms," in *Biometrics: Theory, Applications and Systems (BTAS), 2013 IEEE Sixth International Conference on*. IEEE, 2013, pp. 1–8.

[17] S. Mare, A. M. Markham, C. Cornelius, R. Peterson, and D. Kotz, "Zebra: zero-effort bilateral recurring authentication," in *Security and Privacy (SP), 2014 IEEE Symposium on*. IEEE, 2014, pp. 705–720.

[18] O. Huhta, P. Shrestha, S. Udar, M. Juuti, N. Saxena, and N. Asokan, "Pitfalls in designing zero-effort deauthentication: Opportunistic human observation attacks," *arXiv preprint arXiv:1505.05779*, 2015.

[19] S. Eberz, K. B. Rasmussen, V. Lenders, and I. Martinovic, "Preventing lunchtime attacks: Fighting insider threats with eye movement biometrics." in *NDSS*, 2015.

[20] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*. Springer series in statistics Springer, Berlin, 2001, vol. 1.

[21] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, "A survey on homomorphic encryption schemes: Theory and implementation," *CoRR*, vol. abs/1704.03578, 2017. [Online]. Available: http://arxiv.org/abs/1704.03578