A Usable and Robust Continuous Authentication Framework Using Wearables

Abbas Acar, Hidayet Aksu, A. Selcuk Uluagac, *Member, IEEE*, and Kemal Akkaya, *Senior Member, IEEE*

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. *Continuous authentication*, which re-verifies the user identity without breaking the continuity of the session, can address this issue. However, existing methods for Continuous Authentication are either not reliable or not usable. 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 extensively 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 percent* with 30 seconds of processing time and 2-3 percent for 20 seconds. The computational overhead is minimal. Furthermore, WACA is capable of identifying insider threats with very high accuracy (99.2 percent) and also robust against powerful adversaries such as imitation and statistical attackers. We believe that this work has practical and far-reaching implications for the future of the usable authentication field.

Index Terms—Continuous authentication, wearables, biometrics, keystroke dynamics, typing

1 Introduction

The majority of the current user authentication methods rely on password authentication. However, password authentication methods are subject to many security drawbacks [1], [2], [3]. Many practical attacks have been demonstrated that the passwords can be either stolen or bypassed [4], [5]. To mitigate these threats, Multi-Factor Authentication (MFA) methods were proposed [6], [7], [8]. In MFA, the user credentials are checked from two or more independent sources, and even if the attacker steals one factor, it would still have to overcome the burden of other factors. Though, whether it is one-factor or MFA [8], a one-time login process does not guarantee that the identified user is the real user throughout the login session. Even if it is a legitimate insider who has been authorized once, a forever access is provided in most cases not to interrupt the current user.

An authentication mechanism, which re-verifies the user periodically without breaking the continuity of the session, is vital [9]. For example, users may share their passwords with family members, friends, colleagues [3], 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 environment.

Continuous Authentication (CA).[1] is a good mechanism to re-verify a user identity periodically throughout a login session. However, existing methods are either not reliable or not usable. For instance, currently, the most common method used to verify the user periodically depends on session time-outs. In session time-outs, if the time window is kept too short, the user's convenience will be reduced due to frequent interruptions of the session for authentication. On the other hand, if the time window is set too long, in the case of a breach, the attacker would have more time on the victim's system.

In the literature, a number of studies have been proposed for the use of biometrics in continuous user authentication [11], [12], [13]. However, one of the desired features in continuous authentication is *non-intrusiveness* [14]. Physiological characteristics like iris pattern or fingerprint are not applicable in this manner since they can not be extracted seamlessly. More plausible approaches for CA would be behavioral characteristics [15], [16], [17] like typing rhythm, gait as they can be collected without interrupting the user.

- *A. Acar, H. Aksu, and A.S. Uluagac are with Cyber-Physical Systems Security Lab (CSL), Department of Electrical and Computer Engineering, Florida International University, Miami, FL 33199 USA. E-mail: {aacar001, haksu, suluagac}@fiu.edu.*
- *K. Akkaya is with Advanced Wireless and Security (Adwise) Lab, Department of Electrical and Computer Engineering, Florida International University, Miami, FL 33199 USA. E-mail: kakkaya@fiu.edu.*

Manuscript received 23 Oct. 2018; revised 24 Dec. 2019; accepted 10 Feb. 2020. Date of publication 18 Feb. 2020; date of current version 5 May 2021. (Corresponding author: Abbas Acar.) Digital Object Identifier no. 10.1109/TMC.2020.2974941

1. CA is also sometimes called Active or Implicit Authentication in the literature [10].

Although, behavioral biometrics may suffer from high error rates due to their variability [18]. For example, the behavioral biometrics may have higher false rejections than the physiological biometrics in general. However, they have the advantage of being seamlessly collectible as they can be integrated with the other authentication factors seamlessly. Therefore, they are ideal candidates to increase the security of the current systems as an additional authentication factor rather than a standalone authentication system. For example, the US government has recently launched an initiative to explore the deployment of continuous multi-factor authentication in the office environment [19], where different behavioral biometrics like voice or gait should be utilized to increase assured identity of users.

Among all behavioral biometrics, the most promising results are proposed using keystroke dynamics [20], [21], [22]. However, in a recent work [23], 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 [24]. Indeed, the need for a better authentication mechanism and the emerging widespread availability of wearable devices promise a unique opportunity to utilize the wearable devices to achieve a genuinely usable and practical CA system. The closest work to our approach is proposed in [25] called ZEBRA. In ZEBRA, users are classified according to the sequence of interactions (e.g., typing, scrolling), where the user wears a bracelet with motion sensors and radio. Serious design flaws have been detected in ZEBRA [26]. Our design differs from ZEBRA in many ways to tackle those flaws and strengthen our design. First and most importantly, instead of a sequence of interactions, we use a variant of keystroke dynamics as an authentication factor, which was extensively well-studied in the research literature and shown as a unique behavioral biometric for each person (see [27] for a very recent survey). WACA also has some other design details, which makes it secure against advanced attackers as shown in Section 6.2.

In this work, we introduce a novel *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. Specifically, 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. Note that most conventional keystroke-based authentication schemes in the literature [28] 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-axis motion sensor data can provide not only timing information, but also the key-pressing pressure, hand rotation, and hand displacement, etc. Our feature set consists of 14 different sensory features from both time and frequency domains. These features are applied to 6-axis motion sensor data, obtaining 84 features in total, jointly considering the 6-axis data. 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 [29], [30]. Also, in another work [25], 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 [25] has been shown as insecure in another work [26]. As explained, our work differs from other works in several ways to tackle those flaws and strengthen our design.

We tested the performance, efficiency, and security of WACA with more than thirty real users and data collected from them. We specifically evaluated WACA in terms of three metrics: (i) *How accurately can it authenticate the genuine users and lock out the and impostor users?* (ii) *How fast can it detect an impostor?* (iii) *How accurately can it identify an impostor from its typing pattern?* Moreover, we also evaluated the robustness of our proposed method against powerful attacks, including, *imitation* [23], [26], *statistical* [31], [32], and insider attacks.

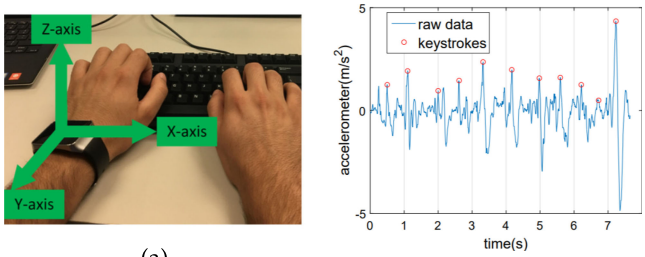
Contributions. The main contributions of this work are summarized as follows:

- We propose a novel sensor-based wearable-assisted continuous authentication framework for computing platforms, terminals (e.g., laptops, computers) with a smartwatch.
- 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 also tested WACA against powerful attacks, including imitation, statistical attacks, and insider attackers. For this purpose, we designed a scenario for the imitation attacks with real participants. On the other hand, we developed three generic attacking scenarios for the statistical attacks that can also be utilized by other future continuous authentication studies.

Organization. The remainder of this paper is structured as follows: Section 2 presents the related work. In Section 3, we explain the foundation for the overall idea. Then, we introduce our system model in Section 4. Then, the overall architecture of WACA is detailed in Section 5. Section 6 presents the performance, efficiency, and robustness evaluation of the WACA framework. Section 8 reports the discussion of the challenges that can be faced in WACA and ways to overcome those challenges. Finally, in Section 9, we conclude the paper.

2 Related Work

In the literature, a number of works have been proposed for the use of biometrics in continuous user authentication [11], [12], [13], [33], [34], [35]. However, one of the desired features in the continuous authentication is *transparency*. Hard biometrics like iris pattern or DNA are not applicable since they can not be extracted transparently. In another work [36], a special mouse with a fingerprint sensor is proposed. In addition to requiring a custom mouse, its reliability is also an issue. The ease of



(a) (b)

Fig. 1. (a) The reference coordinate system for accelerometer and gyroscope sensors. (b) A sample raw data collected from the accelerometer of the smartwatch and keystrokes detected by using peak detection methods while typing the word "smartwatch".

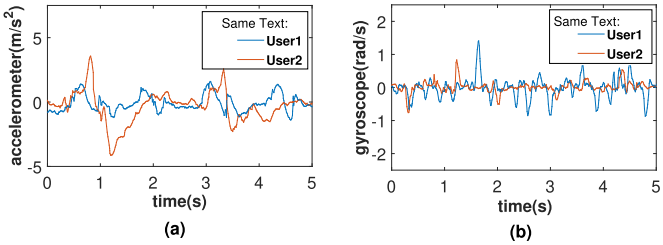


Fig. 2. Comparison of two different users' (a) accelerometer (b) gyroscope readings while typing the same text.

counterfeiting fingerprints was shown, and the fingerprint-based biometrics was easily bypassed [37], [38]. Facial recognition methods may seem a good candidate; however, the liveliness detection is still an issue to be addressed, and several attacks are possible under practical conditions [39], [40]. In addition, several other biometrics like pulse-response [41] or eye movements [42] are also proposed. However, since these approaches require special equipment, deployment costs are increasing significantly.

Recent Suspicions on Keystroke Dynamics. Among all the biometrics, the most promising results are proposed using keystroke dynamics and mouse movements [20], [21], [22]. However, in a recent work [23], the reliability of classical keystroke dynamics are analyzed and an interface, called Mimesis, was designed so that a user can mimic the typing rhythm of another user by using the feedback provided by Mimesis. In another study [31], the statistical attacks with bots generating synthetic typing patterns are examined for the conventional keystrokes biometrics. In our work, we test WACA against both these imitation and statistical attacks using similar configurations presented in these papers. We show that WACA is secure against the powerful imitation and statistical attacks. The detailed analysis of these attacks are given in Section 6.2.

Inference Attacks Using Smartwatch Sensors. Another direction on sensor-based keystroke research is using the motion sensors of wearables as a side channel attack to infer some valuable assets like passwords. The main motivation behind this attack is similar to WACA. Motion sensors will move in the same way with keystrokes while typing and the wrist rotations and displacement will cause to leak the keystrokes. This attack is deployed first on smartphones [43], [44], [45], [46], [47], and recently on smartwatches [48], [49]. Restricting access to motion sensors is not a realistic suggestion to defend against this attack. In our work, we propose a pairing and synchronization session before using the smartwatch with its paired computer. In this way, an encryption-supported secure channel can be used to communicate between smartwatch and computer.

3 Design Rationale: Why Should It Work?

In this section, we study how motion sensors of a smartwatch are impacted when typing on a keyboard and see if the data can be used to identify users. Particularly, we analyze a case that a user wears a smartwatch and types on a qwerty-type built-in keyboard of a computer. Our goal is to collect keystroke information from the built-in motion sensors (i.e., accelerometer and gyroscope) of the smartwatch during the typing activity. To collect smartwatch sensor data, we developed an Android Wear app that records the raw sensor readings from the motion sensors.

In our experiments, we used linear acceleration composite sensor data, which combines the data of accelerometer and gyroscope to exclude the effect of gravity.[2] Note that the accelerometer and gyroscope sensors provide three-dimensional sensor data, where the reference coordinate system associated with the sensors are illustrated in Fig. 1a. As z-axis of the accelerometer sensor is directly affected by the key up-down movements of a user while typing, the most significant changes are observed in the z-axis. Therefore, the z-axis of the data provides the best information for keystroke features such as holding time, pressing pressure, etc. Moreover, another observation is that even if the device is placed flat on a desk, the sensors generate a certain level of noise, which needs to be removed by filtering, as explained later.

Sample data in Fig. 1b was acquired from the z-axis of the accelerometer while typing the word "smartwatch". It can be seen how the value of the accelerometer makes peak points. As the acceleration through the gravity corresponds to the going down of the accelerometer, the peak points in the figure correspond to the keystrokes in the typing activity. While the amplitude of the peak is related to how strong the key press is, the width of the peaks is associated with how long the key is pressed. These are simple statistics that can be used to identify users. These and other features will be further analyzed in detail in Section 5.

Moreover, we conducted two more simple experiments using the accelerometer and gyroscope data on the smartwatch, and we made the following two observations:

- *Observation 1: Different users exhibit different patterns even if they type the same text.*

In this experiment, we compared the data collected from two different users while typing the same text. Fig. 2 presents the sensor data of the two users' accelerometer and gyroscope data for a given time interval. The distribution of the accelerometer data in Fig. 2a shows clear differences such as the magnitude of peaks, inter-arrival time of peak points, the width of peaks, etc. On the other hand, the gyroscope sensor measures the rotation of the watch. As seen in Fig. 2b, the number of peaks or the magnitude of the peaks

2. For brevity, we use acceleration to refer to the linear acceleration.

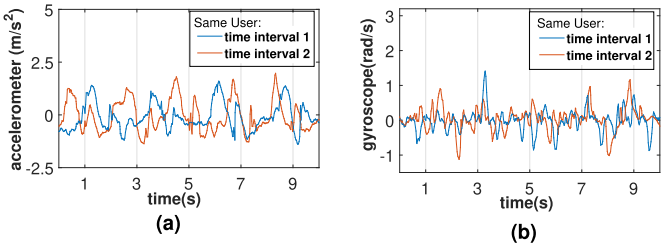


Fig. 3. Comparison of the same user's sensor data over two different time intervals with (a) accelerometer, (b) gyroscope.

are different for different users; so these features are viable candidates to recognize different users.

- *Observation 2: Same user follows similar patterns over different time intervals even while typing different texts.*

In the second experiment, the data was collected from the same user over two different time intervals corresponding to the different texts, and the plots are given in Fig. 3. As seen in Fig. 3a, the amplitudes and widths of the peaks are similar in magnitude, but with a phase shift, meaning leading or lagging. On the other hand, the same leading or lagging of similar shapes can also be seen in the gyroscope data in Fig. 3b.

These two observations justify the rationale that keystroke dynamics obtained from smartwatch accelerometer and gyroscope sensors can differentiate different users as classical keystroke dynamics and the same users can be detected over different times even while typing different texts. Although these are just preliminary observations, our framework will be further tested and evaluated with extensive experiments using real user data in Section 6.

4 System Model

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

Design Goals. In WACA, our design goals is similar to the ones given in [50]: Our system should be *universal* (i.e., the biometric features exist for everyone), *unique* (the features are specific for everyone), *permanent* (the biometric features always exist), *transparent* (the system works without interrupting the user), *continuous* (the system should provide continuous user data), and *accurate* (the system works with low error rate). WACA achieves the first five goals by its design and the accuracy is tested in Section 6.

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 transitions and at rest.
- The WACA framework acts as a complementary second-factor, and it has the flexibility to work any first-factor authentication system, and it is assumed that the system has already a first authentication factor. The first factor could be one of the password-, token-, or biometric-based systems. Note that the first factor of authentication is beyond the scope of this work.

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 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 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 proper timing that its victim leaves the computer unlocked for some time to go to lunch or to get coffee, etc. (aka *lunchtime attack* [42]). 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).
- *More Powerful Adversaries:* Furthermore, a powerful adversary can be aware of WACA and try to defeat it using special tools and skills by *imitating* legitimate users [23], [26] or launching *statistical attacks* [31], [32]. This powerful adversary (insider or outsider) can be a human or a trained bot. In imitation attacks, the attacker wears the victim's smartwatch either via after stealing it, or the victim can leave it behind for a while and the attacker can try to impersonate the victim. On the other hand, the statistical attack is more complex and requires special tools and skills. Hence, WACA also considers these powerful attack scenarios in its adversary model.

The security evaluation of these attack scenarios and how WACA is robust against insiders, imitators, and statistical attackers are explained more in Sections 6.1 and 6.2.

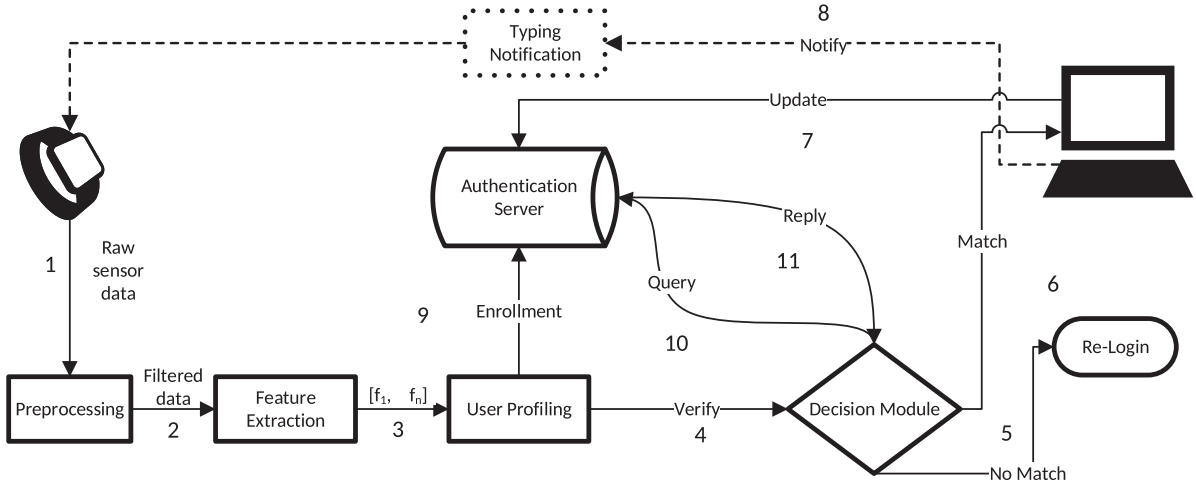


Fig. 4. WACA framework architecture and key components.

5 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.

5.1 Overview

WACA consists of four main stages: *Preprocessing, Feature Extraction, User Profiling, and Decision Module*. These stages, which are shown in Fig. 4, 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) and transformed into a proper format for the next stages.
- 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. An optimum value of this parameter can be set after experimenting with different values in a real-world implementation. 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.

5.2 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 the 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 $\vec{acc} = < t_a, x_a, y_a, z_a >$ and gyroscope is represented as $\vec{gyro} = < t_g, x_g, y_g, z_g >$, 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 the 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.

5.3 Preprocessing

In WACA, *preprocessing* stage refers to the 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. To remove the effect of the noise from data, we apply M-point Moving Average Filter (MAF), which is a simple low-pass filter and it operates by taking the average of M

neighbor points and generates a single output. M-point filtering in equation form can be expressed as follows:

$$\dot{y}[i] = \frac{1}{M}\sum_{j=0}^{M-1} \dot{x}[i+j], \tag{1}$$

where $\dot{x}$ is the raw sensor data, $\dot{y}$ 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., $\vec{x_a} = < {x_a}^1, \ldots, {x_a}^n >$ and $\vec{x_g} = < {x_g}^1, \ldots, {x_g}^n >$ for a profile of n samples.

5.4 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 $\vec{f} = < f_1, f_2, \ldots, f_n >$ 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 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:

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}, \tag{2}$$

where x_{min} and x_{max} are the minimum, and maximum value of the features of the user's enrolled templates.

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 1. These features were chosen as they performed well in similar contexts [29], [30].

5.5 Decision Module

The last 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 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:

$$distance(\vec{x}, \vec{y}) = \left(\sum_{i=1}^{n} (x_i - y_i)^p\right)^{\frac{1}{p}}, \tag{3}$$

where $\vec{x} = (x_1, x_2, \ldots, x_n)$ and $\vec{y} = (y_1, y_2, \ldots, y_n)$ 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 $distance \geq threshold$), the impact of which is analyzed in Section 6.1. Indeed, this threshold measures the confidence of the decision for a given user.

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 shows 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 6.1.

TABLE 1
Feature Set Extracted From Sensor Data in WACA

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

6 Performance Evaluation

We tested the performance, efficiency, and security of WACA with more than thirty 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 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 would be in identifying insider threats.

Data and Collection Methodology. In our experiments, we collected data from 34[3] human subjects.[4] During the

3. Not all of them participated in all experiments.

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

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. The participants were free to choose the hand (left/right) on which they wore the smartwatch. The choice of the hand that the participants wore the smartwatch was left to the participants. Moreover, they were also given the freedom to adjust the sitting position and the keyboard and screen position according to their comfort levels. It is also worth noting that sitting and wrist position (i.e., if it is resting on the table or maintained in the air) may affect the performance. Therefore, a real-world implementation may require further calibration before enrolling the users to the system.

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:* 20 participants are involved in this task, and the participants were asked to type a story from a set of short and simple stories from the American Literature[5] for four minutes. The story was chosen randomly by the participants. On average, four minutes of data corresponds to 25,000 samples for each participant (Total: 850,000 samples).
- *Typing Task-2:* 20 participants are involved in this task and for this data set, all the participants were asked to type the same text[6] for four minutes. For each participant, almost the same amount of data is collected as Typing Task-1. This dataset is essential to be able to measure the quality of the features.
- *Typing Task-3:* 34 participants are involved in this task, and the participants were instructed to imitate someone else' typing pattern by watching the prerecorded video of the other person. For these experiments, one of the participants was recorded on video while typing a short and simple sentence for 15 seconds from a perspective that the hand motions, smartwatch, keyboard, and the screen could be seen. Although it was not required, the perspective allowed to infer what the victim was typing by watching. This dataset was primarily used to analyze the attacking scenarios.

Note that in all the experiments, the dataset obtained from all these tasks were always used by dividing them into equal size chunks. Therefore, even if all the participants in Typing Task-2 typed the same text, the compared samples always corresponded to different texts for a participant.

Moreover, 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 timestamp, and the others are 6-dimensional sensor data. The 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 $100\ Hz$.

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 unauthorized attempts: The increase in FAR is a direct threat to the system's security level (i.e., confidence level on the decision). 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 the threshold. Indeed, the threshold value effectively measures the confidence of the decision for a given user. 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.

6.1 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: *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) dataset and we fixed the sample size to 1,000 and used Manhattan (City-block) as a representative distance measure to compare the samples. Note that as later shown and analyzed in Figs. 7 and 8, this distance metric was chosen as it performed the best among the different distance measurement techniques. This is because Manhattan is rectilinear distance, considering the absolute differences and is more suitable for natural settings [51], [52], [53]. For each sample of a particular user, we computed the differences from other users' samples. For this purpose, we computed the $N \times N$ 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[7] method [54].

Then, for a given threshold and participant, the ratio of the rejected and accepted samples was 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

5. https://americanliterature.com/100-great-short-stories
6. https://en.wikipedia.org/wiki/The_Adventures_of_Tom_Sawyer

7. Even though to show the feasibility of our method, we tested our method with leave-one-out cross-validation, collecting and storing more than one samples from each user at the enrollment phase may impact the accuracy in real-life implementations.

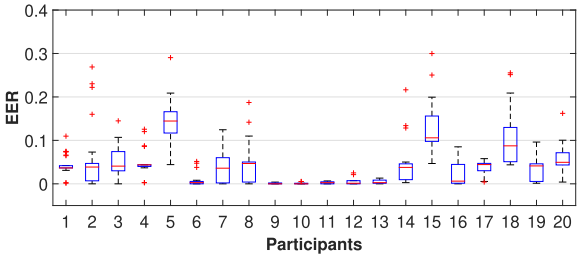


Fig. 5. EER for each participant with a sample size of 1000 using Manhattan (Cityblock) distance metric during Typing Task-1. Average EER is 0.0513.

WACA. The results are plotted in Fig. 5 for Typing Task-1 and Fig. 6 for Typing Task-2. Average EER for the Typing Task-1 experiment was 0.0513. Fig. 6 compares the EER of participants for the Typing Task-2 experiment. Average EER for this experiment was 0.0647.

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 behaviors (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 3,000 and utilized five different distance measuring techniques, euclidean (p = 2), Cosine, Correlation, Cityblock, and Minkowski (p = 5). Again, two types of participant dataset, Typing Task-1 (any text) and Typing Task-2 (the same text), were used. Fig. 7 (Typing Task-1) and Fig. 8 (Typing Task-2) present the main results when the sample size increases.

As can be seen in Fig. 7 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 1,500 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 3,000. In the best case, EER 0.007 is achieved with the

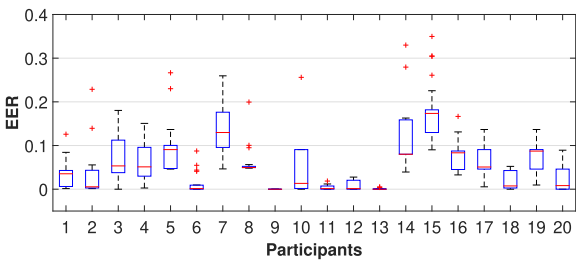


Fig. 6. EER for each participant with a sample size=1000 using Manhattan (Cityblock) distance metric during Typing Task-2. Average EER is 0.0647.

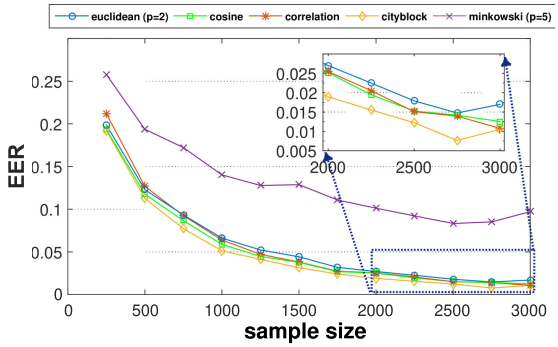


Fig. 7. Average EER according to different sample sizes using different distance metrics while users are performing Typing Task-1.

sample size of 2,750 for the Manhattan (i.e., Cityblock) distance measurement technique.

Fig. 8 presents the results of the same-text experiment (Typing Task-2). As in Fig. 7, 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 3,000. We also see the convergence of EER in Fig. 8 as Fig. 7. Plots are starting to converge around sample sizes 1,500-2,000 and converging to 0.01 for Cityblock and Correlation distance measuring techniques. We also see that at 3,000, 0.02 EER is obtained for Cosine and Correlation techniques. However, if shorter data collection time is of interest, a sample size of 2,000, 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 security policies.

To conclude, the features in WACA can successfully differentiate the users from their typing rhythm with a minimal error rate (1 percent) independent of the typed text. There is an inherent 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. To effectively locate such an insider attacker within an organization where WACA is employed, an

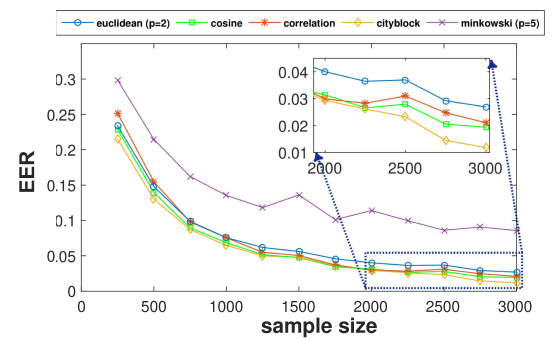


Fig. 8. Average EER according to different sample sizes using different distance metrics while users are performing Typing Task-2.

TABLE 2
Evaluation of the Insider Threat Identification Results With Seven Different Machine Learning Algorithms

Classifier	Typing Task-1	Typing Task-2
SVM	98.7	98.1
Random Forest	98.9	97.8
Naive Bayes	93.6	87.3
Decision Tree	62.1	62.1
MLP	**99.0**	**99.2**
kNN	96.4	96.8
Logistic Regression	90.5	93.7

MLP yields the best result and the training/validation graphs of the MLP algorithm are given in Appendix B.2, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TMC.2020.2974941.

identification mechanism is needed. Depending on the security policy of the organization, the management may want to do an investigation to find the insider attacker. In this case, we will have many unknown samples of the attacker to find the owner of the samples, and we will need a one-to-many classification task to exactly detect an insider attacker. For this purpose, we fix the sample size to 1,500 and the number of training sample to five. With those parameters, we tested different machine learning algorithms and results are presented in Table 2. Here, we assume that the insider's data is also stored in the authentication server's database (training set) as a legitimate user.

According to the results given in Table 2, the most accurate results are obtained with the Multilayer Perceptron (MLP) algorithm. This happens because of two reasons. First, MLP is a neural network model, which maps a set of input data into a set of outputs through the interconnected processing elements (neurons). The main advantage of MLP is that it approximates highly nonlinear functions between input and output [55]. Second, when we look at literature, MLP is giving very high accuracy with the features obtained from noisy sensor data collected from devices like smartphone or smartwatch. We present these works as well as all the technical details of MLP used in our work in Table 7 in Appendix B, available in the online supplemental material.

We also analyzed the impact of the sample size and the size of the training data on the accuracy. For this, we focused on two test scenarios that would be relevant in real investigations and tested the efficacy of 7 different machine learning algorithms. As seen in Table 2, MLP performed the best and accordingly we picked MLP as a representative algorithm to be used in these scenarios: *Scenario 1:*[8] 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 requested again to type the same text. The results are presented in Table 3. *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 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 3.

TABLE 3
The Accuracy Results Insider Threat Identification Experiments for Different Sample Sizes in Scenario 1 and 2

Sample size	Training Set				
	1	2	3	4	5
Scenario 1: Accuracy (%)					
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 (%)					
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

As can be seen in Table 3, in the best case, 99.2 percent identification rate of an insider threat can be achieved with the sample size of 1,500 while the model is trained with five samples. Even with two samples of the insider, 93.7 percent accuracy rate can be achieved with the sample size of 1,500.

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 3 presents the result of this question for Scenario 2. As can be seen from Table 3, 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. Three training samples and the sample size is 1,500 or four training samples with the sample size of 1,000 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 Figs. 7 and 8) 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 five training samples with 1500 sample sizes are obtained from a potential insider threat, WACA could identify the insider with 99.2 percent accuracy rate while typing the provided text (see Table 3) or with 91.8 percent accuracy rate while typing a random text (see Table 3).

6.2 Advanced Attacks on WACA With More Powerful Adversaries

In this subsection, we evaluate the performance of WACA against two powerful attacks: imitation [23], [26] and statistical [31], [32] attacks. In these attacks, the attacker is aware of WACA and can try to defeat WACA using special tools and skills.

Numerous attacks against classical keystroke dynamics that exist in the literature can also be used to attack WACA. The attacker can be a human or a trained bot. A human-type attacker can perform *zero-effort attacks*[9] [56] or *imitation*

8. Please note that this is different the Typing Task-1 in Fig. 2.

9. Also called *zero-information attack*.

attacks [23] to defeat the WACA's authentication system. In *zero-effort attacks*, the attacker tries to defeat the authentication system without any effort or prior knowledge. Zero-effort attacks will not be successful due to the low EER values in WACA as analyzed in the previous sub-sections. However, the effectiveness of the imitation attacks performed by a human should be investigated as noted in some recent studies [23], [26].

In addition to these attacks, another recent attack against the behavioral biometrics [31], [32] has emerged, which is called *statistical attacks*. In this attack, a bot is first trained using typical user data from a large population. Then, the bot generates random permutations of the features to mimic a legitimate user. In addition to human and robot attacks, a *replay attack* using a key-logger [57] is noted in the literature, which can also be performed against the keystroke dynamics. However, a key-logger installed on the computer can obtain only the timing of the keystrokes, which is solely not enough to use it in a replay attack against WACA as there is not a way that a key-logger can obtain the three-dimensional sensor data collected by the smartwatch.

In the next sub-sections, we consider these two powerful attacks (imitation and statistical) and investigate the effectiveness of WACA against them. In these cases, the attacker would have somehow obtained the victim's smartwatch or manipulates his smartwatch. We use the zero-effort attacks as a baseline to evaluate the success of the imitation and statistical attacks. In imitation attacks, the attacker either can steal the victim's smartwatch or the victim can leave it behind for a while, then the attacker wears the victim's smartwatch and can try to impersonate him while attacking. On the other hand, the statistical attack is more complex and requires special tools and skills. In this type of attack, we assume the attacker can provide its input data to the system. It manipulates its username and profile data to get access to the computer that he does not have permission.

6.2.1 Imitation Attacks

In this subsection, we evaluate the performance of an imitating adversary, who knows that WACA is already installed on the current system. The adversary is assumed to be watching his victim by standing nearby or trying to imitate the victim's typing style by looking at the previously recorded video of the victim while typing. S/he is also assumed to be opportunistically waiting for the right time to mimic the victim.

To replicate this imitation attack scenario, we recorded a 15 seconds video of a legitimate user and presented this video to an attacker (i.e., another participant in our experiments). The video showed the user as s/he was typing and thus the hand, fingers, watch and keyboard were all visible. By watching the video (multiple times allowed in experiments), the attacker tried to imitate the legitimate user. Note that this scenario would increase the chances of a successful attack when compared to a real-life case where the attacker would possibly only have limited opportunity to watch a victim. We also collected the victim's typing data to evaluate the performance of the attackers. We computed EER for this attack scenario and compared it with the case when there was a zero-effort for the attack. In the zero-effort attack, we used the data set obtained in Typing Task-1 Dataset. We applied the leave-one-out method [54] by leaving the victim's data out as in the other authentication experiments While calculating EER (i.e., the intersection of FAR and FRR) of the victim. In the imitation attack, since we only had the impostor attempts, EER would be equal to the attacker's acceptance rate. We also note that WACA was directly tested without any change. The results are presented in Fig. 9.

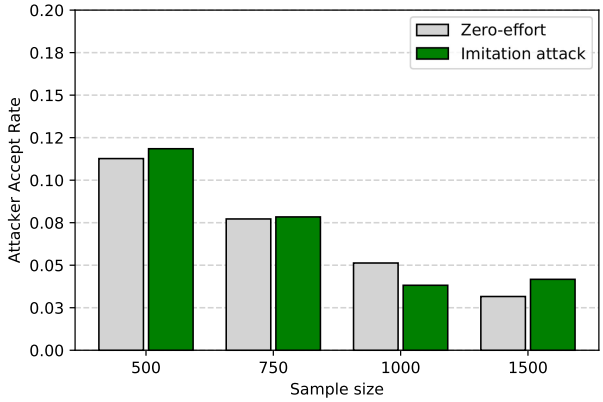


Fig. 9. Attacker accept rates for different sample sizes. The results show that an imitation attacker has no more advantage than a zero-effort attacker.

As presented in Fig. 9, the attackers have different success rates (attacker accept rate) for different sample sizes. The highest success rate was achieved when the sample size is equal to 500, but the success rates are decreasing to the much lower rates as the sample sizes increase. A sample size of 500 corresponds to almost 2-3 keystrokes for the sampling rate used, which is not enough to measure and settle down for some of the features. So, this is not practical from the attacker's perspective. Beyond 1,500, which corresponds to 15 seconds of sensor readings, the probability of an attacker to imitate a user is significantly decreasing (i.e., 0.04). *These results indicate that even though an attacker is aware of WACA in a targeted system, s/he still has a meager chance to be successful.*

6.2.2 Statistical Attacks

In this subsection, we evaluate WACA against statistical attacks. In this attack scenario, it is assumed that the attacker has a database obtained from AS consisting of the user profiles. Similar to the imitation attack, it is also assumed that the attacker can provide its input to the system. As mentioned earlier, this can occur either by obtaining the victim's smartwatch or if the attacker is an insider, it can manipulate its input data to deceive WACA. It is also worth mentioning that we assume the attacker has only a limited amount of time to attack; therefore, it only tries the data that has the highest chance to get in, which we refer to as $topBins$ in the attack algorithm that will be utilized and noted below.

Note that statistical attacks are very powerful attacks and it is successfully implemented to bypass the conventional keystroke-based systems [31]. It is based on the generation of fake (synthetic) inputs using common features of a given population. The idea behind this attack is using the random combination of the most common features of the population to defeat the authentication system. We designed the following attack scenario to test WACA against the statistical attacks.

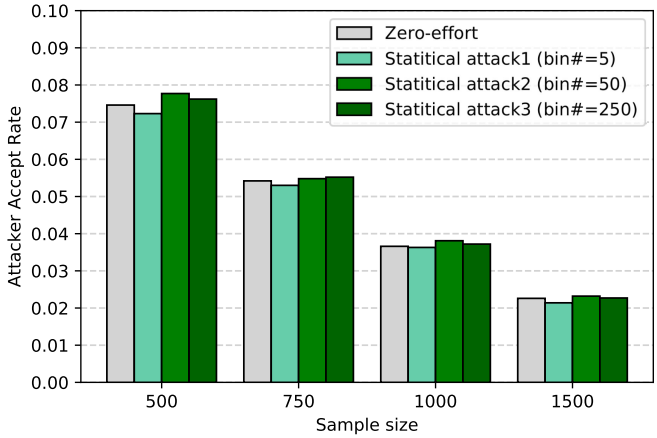


Fig. 10. 3 different statistical attacks against WACA with different sample sizes.

TABLE 4
Resource Consumption of the Smartwatches Used in the Experiments: *LG Watch R* and *Samsung Gear Live*

		LG G Watch R	Samsung Gear Live
CPU (no WACA)		4.5%	4.5%
CPU (w/WACA)		7.5%	16.8%
Memory (no WACA)		4.5 MB	4.5 MB
Memory (w/WACA)		15.2 MB	13.8 MB
Battery	10 Hz	1.1%	1.2%
	30 Hz	1.6%	0.3%
	100 Hz	2.1%	2.4%
Data Size	10 Hz	0.3 MB	0.3 MB
	30 Hz	0.6 MB	0.9 MB
	100 Hz	4.1 MB	6.5 MB

In our attack, we used both Typing Task-1 and Typing Task-2 dataset as input. Each participant was chosen as a victim iteratively, and the other participants' samples were used to generate forged data samples. Then, the forged samples were used to attack the victim. For this, a histogram was created for each feature of all the participants in the dataset except the victim. The forged samples were generated as in Algorithm 1 in Appendix A, available in the online supplemental material. Overall, we created three different statistical attackers with three different capabilities (bin sizes in the histogram). Statistical Attacker-1, Statistical Attacker-2, and Statistical Attacker-3. Before running the algorithm for attacking WACA, we first calculated the EER for each user without adding any forged data. Similar to the imitation attacks, the attacker acceptance rate in zero-effort attack corresponds to the average EER. We conducted experiments without attack under varying sample and bin sizes. The results are shown in Fig. 10.

In Fig. 10, we can see that bin number 50 has the most successful result on attacking victims. This is because if we increase the bin number and keep the bins with the highest number of occurrences constant, the width of the bins will narrow; so, the range of the forged data will be confined to a very small range. On the other hand, if we decrease the bin number significantly, the less frequently occurred bins will also be included in the sample generation range, which will reduce the success rate of the attacks. Finally, we note that in the attack scenario, we choose each user in our dataset as a victim in an iterative way. *These results show that despite the small increase compared to zero-effort, the attacker does not have a chance to defeat WACA using the systematically generated fake data due to its high dimensional feature vector in WACA's design.*

As a summary, neither the imitation nor statistical attacks put WACA in danger as their success rates are very close to zero-effort attacks.

6.3 Resource Consumption

In WACA, a smartwatch, a computer, and an authentication server work together. In this subsection, we only analyze the resource consumption of relatively constrained smartwatches. It is worth noting that we monitored the consumption of our application while it was running continuously; however, in WACA, the data collection app does not have to be running continuously. It can happen periodically or on-demand because the data collection runs only when the smartwatch is notified by the computer that the user is typing on. We analyze the performance of both LG G Watch R and Samsung Gear Live smartwatches used in the experiments. Both smartwatches have Cortex-A7 at 1.2 GHz and 512 MB RAM, but Samsung uses 300 mAh battery, while LG is using 410 mAh battery. The results are presented in Table 4.

In all the experiments, we both monitored the memory and CPU resource utilization of the smartwatches in the default mode (i.e., not actively running any app - no WACA) and while the app was running (w/WACA). In the default mode, both smartwatches used almost 4.5 MB memory and 4.5 percent CPU their consumption while the app was running, as shown in Table 4. As compared to the default memory usage (no WACA), the memory consumption in the smartwatch in WACA is increasing, but it is still at an acceptable rate.

In addition to memory and CPU consumption, we also analyzed the power consumption and data size while running our app for 10 minutes. We excluded the power consumption of the screen as the screen can be turned off or the smartwatch can be in the ambient mode during the data collection of WACA. We see that the power consumption of the app scales by the sampling frequency. However, when we decrease the sampling rate, the time needed to collect a certain amount of data will also increase. Hence, the optimum sampling rate should be tuned according to the desired security policy.

7 Comparative Evaluation

In the literature, there is not a widely accepted standard framework to compare device authenticators. However, Usability-Deployability-Security (UDS) framework proposed in [58] is a highly accepted framework for web authentication schemes. To compare our work with its alternatives, we remove some of the irrelevant and non-applicable benefits and use only the relevant ones of the UDS framework. The complete list of benefits can be found in [58]. After also

TABLE 5
Comparative Evaluation of WACA Using the UDS Framework [58] With Continuous Authentication Alternatives

	Memorywise-Effortless	*Nothing-to-Carry*	*Physically-Effortless*	*Infrequent-Errors*	*Easy-Recovery-from-Loss*	*No-Constraint-on-Using-the-Device*	*Accessible*	*Negligible-Cost-per-User*	*Resilient-to-Physical-Observation*	*Resilient-to-Targeted-Impersonation*	*Resilient-to-Internal-Observation*	*Resilient-to-Leaks-from-Other-Verifiers*	*Resilient-to-Phishing*	*Resilient-to-Theft*	*Requiring-Explicit-Consent*	*Unlinkable*	*Insider-Identification*	*Resilient-to-Insider-Threat*
	Usability						Dep.		Security									
Password	●		●	●	●	●	●	●							●	●		
Time-out	●	●	◐	●	●	●	●	●			na	●		●	na		na	
Proximity [41]	●	◐	◐	◐	◐	●	●	◐	◐	na	●							
Face [42]	●	●	◐			◐	◐	◐	●	◐		●		●	◐	◐	●	●
Fingerprint [26]	●	●	◐				◐		●	◐		●		●	◐	◐	●	●
Eye-movement [30]	●	●	◐				◐		●	●	●	●	●	●	●	●	●	●
Keystroke [43]	●	●	◐	◐		●	●	●	◐		●		●		●			●
ZEBRA [18]	●	◐	◐	◐	◐	●	●	◐			●	●	●	◐	●	●	◐	●
WACA (this work)	●	◐	◐	◐	◐	●	●	◐	●	●	●	●	●	●	●	●	●	●

● = *offers the benefit;* ◐ = *almost offers the benefit; no circle = does not offer the benefit.*

adding three new benefits, we end up with 18 benefits in total. Table 5 rates WACA using these 18 benefits. For space, we cannot compare WACA to all continuous authentication methods proposed in the literature. Therefore, we choose representatives for each continuous authentication method.

WACA captures the sensor readings through a smartwatch without interrupting the user, i.e., unobtrusively. However, unlike time-out or classical keystroke dynamics, it requires an extra channel to collect data, but obviously a smartwatch is a not a customized hardware, i.e., it is an off-the-shelf device, so we say it partially supports the benefit of *Nothing-to-Carry* and since its error is deficient, it also offers the advantage of *Infrequent-Errors*. On the other hand, WACA outperforms all other methods in terms of security benefits. In addition to WACA, eye-movement based authentication method also seems as secure as WACA. However, WACA's performance for usability and deployability is better. For example, WACA offers much lower error rates, and eye-movement based methods require a specialized eye or gaze-trackers and the user should be in a certain distance and in front of the eye tracker which obstructs the usability of the eye-movement based methods. They are more convenient for challenge-response type authentication methods [62] even though they have the capability to provide data continuously and transparently. In brief, our conclusion from this comparative evaluation shows that WACA offers better security benefits while keeping the usability at the same level as other notable methods.

8 Discussion

Security Policy Implementation Considerations. WACA works by checking if the current user's profile matched the profile of the logged-in user. When an unauthorized access attempt is detected, the reaction depends on the previously decided security policy. Depending on the security policy, when an attacker is detected, the screen can be locked, and the user can be challenged to re-login; the management and security teams can be alerted in real-time, or a notification e-mail can be sent to the registered e-mail of the logged-in user, and so on. Moreover, we showed that WACA could differentiate an insider from an outsider accurately. In suspicious cases, the administrator can do further investigation to detect the insider, and as we noted earlier, the insider detection is possible in WACA. We also note that even if WACA catches an insider attacker, WACA can not know if the attacker has the full key, which is out of scope this work. Therefore, even if the system is logged-out, an insider can log-in again if it has the full key. Therefore, resetting the initial authentication factor should be considered in the security policy in this case. Finally, the server can also log the failed attempts to prevent attacks aiming to drain the smartwatch's battery.

Moreover, if WACA is deployed in an environment where typing is not required much, the actions that will be performed when the user is not typing should be defined in the security policy. A straightforward solution to this problem can be reducing the system to default security, i.e., locking out the user if there is inactivity for a certain duration.

WACA captures the typing patterns of the user only from one wrist. If the wrist wearable is on the left hand, for example, the typing pattern for the words "and" and "aod" would be the same. This can be perhaps exploited by the attacker by using the letters on the right. However, this would be a remote possibility. In WACA, we wanted to test our proposed method in a more realistic scenario assuming a user will wear a wearable on both hands might be an unrealistic assumption. However, in highly extreme cases, i.e., highly critical environments, two smartwatches can be utilized to collect data from two hands of the users. This will prevent against this type of attack. This should be considered while deploying WACA in a real-world application.

Privacy. In WACA, the computer and the wearable are the devices that belong to the user or belong to the same authentication realm and thus are trusted. The only device that may threaten privacy is the AS. As for the security of the data at rest at the server, the existing industry standards such as AES, RSA, ECC, RC4, can be employed to establish the security of the data in these cases. In WACA, after collecting the raw sensor data from the smartwatch, either the raw sensor data can be transmitted to the AS, or the features can be computed on the smartwatch and the feature vector can be transmitted.No data is stored on the watch and as noted in the Assumptions Section (Section 4), this channel is secured with existing methods. If the raw sensor data is sent to the AS, the AS may try to infer the user's typed characters from the raw sensor data. The more secure way would be to compute the features on the smartwatch and to keep the feature vectors of the profiles of the users in the AS. In that case, the transmitted feature vector has only the mean of the values of the multi-dimensional sensor data and thus inferring the typed characters would not be possible at the AS.

9 Conclusion

Wearables such as smartwatches and fitness trackers have grown exponentially in a short period of time. It is estimated that one in four smartphone owners will also be using a wearable device such as a smartwatch. This ubiquity of wearable devices makes them a perfect candidate to utilize in continuous authentication settings. The continuous

authentication allows users to be re-verified periodically throughout the login session without breaking the continuity of the session. In this paper, we introduced a novel Wearable-Assisted Continuous Authentication (WACA) utilizing the sensory data from the built-in motion sensors available on smartwatches. WACA is a practical and usable wearable-assisted continuous authentication system that combines the functionality of wearables and usability of continuous authentication. 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. We evaluated the efficacy and robustness of WACA with real data from real experiments. The results showed that WACA could achieve 1 percent EER for 30 seconds or 2-3 percent EER for 20 seconds of data collection time and error rates are as low as 1 percent with almost a perfect (99.2 percent) insider threat identification rate.

Acknowledgments

This work was 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] J. Bonneau, C. Herley, P. C. van Oorschot, and F. Stajano, "Passwords and the evolution of imperfect authentication," *Commun. ACM*, vol. 58, no. 7, pp. 78–87, 2015.
[2] E. Grosse and M. Upadhyay, "Authentication at scale," *IEEE Security Privacy*, vol. 11, no. 1, pp. 15–22, Jan./Feb. 2013.
[3] Kaspersky, "Consumer security risks survey 2016," 2016. [Online]. Available: https://dl.acronis.com/u/pdf/Kaspersky_B2C_survey_2016_report.pdf
[4] B. Dickson, "The many ways your password can be stolen or bypassed," Feb. 2016. [Online]. Available: https://bdtechtalks.com/2016/02/12/
[5] M. Theofanos, S. Garfinkel, and Y.-Y. Choong, "Secure and usable enterprise authentication: Lessons from the field," *IEEE Security Privacy*, vol. 14, no. 5, pp. 14–21, Sep./Oct. 2016.
[6] S. Dispensa, "Enhanced multi factor authentication," U.S. Patent 9,762,576, Sep. 12, 2017.
[7] M. Sajjad *et al.*, "Cnn-based anti-spoofing two-tier multi-factor authentication system," *Pattern Recognit. Lett.*, vol. 126, pp. 123–131, 2019.
[8] A. Acar, W. Liu, R. Bayeh, K. Akkaya, and A. S. Uluagac, "A privacy-preserving multifactor authentication system," *Secur. Privacy*, vol. 2, no. 5, 2019, Art. no. e88.
[9] Google, "Google eyes behavioural solution for continuous authentication," May 2016. [Online]. Available: http://www.planetbiometrics.com/article-details/i/4512/
[10] Active authentication, 2013. [Online]. Available: http://www.darpa.mil/program/active-authentication
[11] 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.
[12] Z. Sitova *et al.*, "HMOG: New behavioral biometric features for continuous authentication of smartphone users," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 5, pp. 877–892, May 2016.
[13] H. Feng, K. Fawaz, and K. G. Shin, "Continuous authentication for voice assistants," in *Proc. 23rd Annu. Int. Conf. Mobile Comput. Netw.*, 2017, pp. 343–355.
[14] D. Dasgupta, A. Roy, and A. Nag, *Advances in User Authentication*. Berlin, Germany: Springer, 2017.
[15] I. C. Stylios *et al.*, "A review of continuous authentication using behavioral biometrics," in *Proc. SouthEast Eur. Des. Autom. Comput. Eng. Comput. Netw. Soc. Media Conf.*, 2016, pp. 72–79.
[16] S. Eberz, K. B. Rasmussen, V. Lenders, and I. Martinovic, "Evaluating behavioral biometrics for continuous authentication: Challenges and metrics," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, 2017, pp. 386–399.
[17] 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, 2018, Art. no. 179.
[18] A. K. Jain, A. Ross, and S. Prabhakar, "An introduction to biometric recognition," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, no. 1, pp. 4–20, Jan. 2004.
[19] Disa makes strides toward next generation of authentication, 2018. [Online]. Available: https://www.disa.mil/NewsandEvents/2018/DISA-Common-Access-Card-replacement
[20] M. Antal, L. Z. Szabó, and I. László, "Keystroke dynamics on Android platform," *Procedia Technol.*, vol. 19, pp. 820–826, 2015.
[21] C. Wu *et al.*, "Keystroke dynamics enabled authentication and identification using triboelectric nanogenerator array," *Materials Today*, vol. 21, no. 3, pp. 216–222, 2018.
[22] J. Chen *et al.*, "Personalized keystroke dynamics for self-powered human–machine interfacing," *ACS Nano*, vol. 9, no. 1, pp. 105–116, 2015.
[23] C. M. Tey, P. Gupta, and D. Gao, "I can be you: Questioning the use of keystroke dynamics as biometrics," in *Proc. 20th Annu. Netw. Distrib. Syst. Secur. Symp.*, 2013, pp. 1–17.
[24] A. K. Sikder, H. Aksu, and A. S. Uluagac, "6thSense: A context-aware sensor-based attack detector for smart devices," in *Proc. 26th USENIX Conf. Secur. Symp.*, 2017, pp. 397–414.
[25] S. Mare, A. M. Markham, C. Cornelius, R. Peterson, and D. Kotz, "ZEBRA: Zero-effort bilateral recurring authentication," in *Proc. IEEE Symp. Security Privacy*, 2014, pp. 705–720.
[26] O. Huhta, P. Shrestha, S. Udar, M. Juuti, N. Saxena, and N. Asokan, "Pitfalls in designing zero-effort deauthentication: Opportunistic human observation attacks," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2016, pp. 1–14.
[27] M. L. Ali, J. V. Monaco, C. C. Tappert, and M. Qiu, "Keystroke biometric systems for user authentication," *J. Signal Process. Syst.*, vol. 86, pp. 175–190, 2017.
[28] P. S. Teh, A. B. J. Teoh, and S. Yue, "A survey of keystroke dynamics biometrics," *The Sci. World J.*, vol. 2013, 2013, Art. no. 408280.
[29] K. S. Killourhy and R. A. Maxion, "Comparing anomaly-detection algorithms for keystroke dynamics," in *Proc. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, pp. 125–134.
[30] A. Serwadda, V. V. Phoha, and Z. Wang, "Which verifiers work?: A benchmark evaluation of touch-based authentication algorithms," in *Proc. IEEE 6th Int. Conf. Biometrics: Theory Appl. Syst.*, 2013, pp. 1–8.
[31] A. Serwadda and V. V. Phoha, "Examining a large keystroke biometrics dataset for statistical-attack openings," *ACM Trans. Inf. Syst. Secur.*, vol. 16, no. 2, 2013, Art. no. 8.
[32] V.-D. Stanciu, R. Spolaor, M. Conti, and C. Giuffrida, "On the effectiveness of sensor-enhanced keystroke dynamics against statistical attacks," in *Proc. 6th ACM Conf. Data Appl. Secur. Privacy*, 2016, pp. 105–112.
[33] C. M. Carrillo, "Continuous biometric authentication for authorized aircraft personnel: A proposed design," Master's thesis, Naval Postgraduate School Monterey, CA, Jun. 2003.
[34] H.-B. Kang and M.-H. Ju, "Multi-modal feature integration for secure authentication," in *Proc. Int. Conf. Intell. Comput.*, 2006, pp. 1191–1200.
[35] A. Azzini, S. Marrara, R. Sassi, and F. Scotti, "A fuzzy approach to multimodal biometric continuous authentication," *Fuzzy Optim. Decis. Making*, vol. 7, no. 3, pp. 243–256, 2008.
[36] G. Kwang, R. H. Yap, T. Sim, and R. Ramnath, "An usability study of continuous biometrics authentication," in *Proc. Int. Conf. Biometrics*, 2009, pp. 828–837.
[37] Chaos computer club, fingerprint recognition at the supermarket as insecure as biometrics in passports, 2007. [Online]. Available: https://ccc.de/en/updates/2007/umsonst-im-supermarkt
[38] Chaos computer club breaks apple touchid, 2013. [Online]. Available: http://www.ccc.de/en/updates/2013/ccc-breaks-apple-touchid/
[39] N. M. Duc and B. Q. Minh, "Your face is not your password face authentication bypassing Lenovo–Asus–Toshiba," *Black Hat Briefings*, 2009. [Online]. Available: https://blackhat.com/presentations/bh-dc-09/Nguyen/BlackHat-DC-09-Nguyen-Face-not-your-password.pdf
[40] A. Boehm *et al.*, "SAFE: Secure authentication with face and eyes," in *Proc. Int. Conf. Privacy Secur. Mobile Syst.*, 2013, pp. 1–8.

[41] K. B. Rasmussen, M. Roeschlin, I. Martinovic, and G. Tsudik, "Authentication using pulse-response biometrics," in *Proc. Annu. Netw. Distrib. Syst. Secur.*, 2014, pp. 1–14.
[42] S. Eberz, K. B. Rasmussen, V. Lenders, and I. Martinovic, "Preventing lunchtime attacks: Fighting insider threats with eye movement biometrics," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2015, pp. 1–13.
[43] E. Miluzzo, A. Varshavsky, S. Balakrishnan, and R. R. Choudhury, "Tapprints: your finger taps have fingerprints," in *Proc. 10th Int. Conf. Mobile Syst. Appl. Services*, 2012, pp. 323–336.
[44] E. Owusu, J. Han, S. Das, A. Perrig, and J. Zhang, "ACCessory: Keystroke inference using accelerometers on smartphones," *Proc. 12th Workshop Mobile Comput. Syst. Appl.*, 2012, Art. no. 9.
[45] L. Cai and H. Chen, "TouchLogger: Inferring keystrokes on touch screen from smartphone motion," in *Proc. 6th USENIX Conf. Hot Topics Secur.*, 2011, vol. 11, p. 9.
[46] A. J. Aviv, B. Sapp, M. Blaze, and J. M. Smith, "Practicality of accelerometer side channels on smartphones," in *Proc. 28th Annu. Comput. Secur. Appl. Conf.*, 2012, pp. 41–50.
[47] Z. Xu, K. Bai, and S. Zhu, "TapLogger: Inferring user inputs on smartphone touchscreens using on-board motion sensors," in *Proc. 5th ACM Conf. Secur. Privacy Wireless Mobile Netw.*, 2012, pp. 113–124.
[48] H. Wang, T. T.-T. Lai, and R. Roy Choudhury, "MoLe: Motion leaks through smartwatch sensors," in *Proc. 21st Annu. Int. Conf. Mobile Comput. Netw.*, 2015, pp. 155–166.
[49] C. X. Lu *et al.*, "Snoopy: Sniffing your smartwatch passwords via deep sequence learning," *Proc. ACM Interactive Mobile Wearable Ubiquitous Technol.*, vol. 1, no. 4, 2018, Art. no. 152.
[50] A. Jain, R. Bolle, and S. Pankanti, *Biometrics: Personal Identification in Networked Society*. Berlin, Germany: Springer, 2006.
[51] M. Kokare, B. Chatterji, and P. Biswas, "Comparison of similarity metrics for texture image retrieval," in *Proc. Conf. Convergent Technol. Asia-Pacific Region*, 2003, vol. 2, pp. 571–575.
[52] N. D. Phung, M. M. Gaber, and U. Rohm, "Resource-aware online data mining in wireless sensor networks," in *Proc. IEEE Symp. Comput. Intell. Data Mining*, 2007, pp. 139–146.
[53] Q. Bao and P. Guo, "Comparative studies on similarity measures for remote sensing image retrieval," in *Proc. IEEE Int. Conf. Syst., Man Cybern.*, 2004, vol. 1, pp. 1112–1116.
[54] J. Friedman, T. Hastie, and R. Tibshirani, *The Elements of Statistical Learning*, vol. 1. Berlin, Germany: Springer, 2001.
[55] M. W. Gardner and S. Dorling, "Artificial neural networks (the multilayer perceptron)–A review of applications in the atmospheric sciences," *Atmospheric Environ.*, vol. 32, no. 14, pp. 2627–2636, 1998.
[56] A. Rattani and N. Poh, "Biometric system design under zero and non-zero effort attacks," in *Proc. Int. Conf. Biometrics*, 2013, pp. 1–8.
[57] C. Giuffrida, S. Ortolani, and B. Crispo, "Memoirs of a browser: A cross-browser detection model for privacy-breaching extensions," in *Proc. 7th ACM Symp. Inf. Comput. Commun. Secur.*, 2012, pp. 10–11.
[58] J. Bonneau, C. Herley, P. C. Van Oorschot, and F. Stajano, "The quest to replace passwords: A framework for comparative evaluation of web authentication schemes," in *Proc. IEEE Symp. Secur. Privacy*, 2012, pp. 553–567.
[59] M. D. Corner and B. D. Noble, "Zero-interaction authentication," in *Proc. 8th Annu. Int. Conf. Mobile Comput. Netw.*, 2002, pp. 1–11.
[60] K. M. Beunder, "Design of continuous authentication using face recognition," in *Proc. 20th Twente Student Conf. IT*, 2014, pp. 1–8.
[61] F. Monrose and A. Rubin, "Authentication via keystroke dynamics," in *Proc. 4th ACM Conf. Comput. Commun. Secur.*, 1997, pp. 48–56.
[62] I. Sluganovic, M. Roeschlin, K. B. Rasmussen, and I. Martinovic, "Using reflexive eye movements for fast challenge-response authentication," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 1056–1067.

Abbas Acar received the BS degree in computer engineering from Middle East Technical University, Ankara, Turkey, in 2015, and the MS degree in electrical engineering from the Department of Electrical and Computer Engineering, Florida International University, Miami, Florida, in 2019. He is currently working toward the PhD degree from the Department of Electrical and Computer Engineering, Florida International University, Miami, Florida. He is currently a graduate research assistant with the Department of Electrical and Computer Engineering, Florida International University, Miami, Florida. His research interests include continuous authentication, and IoT security/privacy, and homomorphic encryption. For more information, please visit http://web.eng.fiu.edu/aacar001/.

Hidayet Aksu received the PhD degree from the Department of Computer Engineering, Bilkent University, Ankara, Turkey, in 2014. He is now at Google, prior to that he worked at Amazon. He conducted research as postdoctoral associate at Florida International University (FIU), Miami, Florida, in 2015–2018 and as visiting scholar at IBM T. J. Watson Research Center, Ossining, New York, in 2012–2013. He also worked as an adjunct faculty with the Computer Engineering Department, Bilkent University, Ankara, Turkey and at Scientific and Technological Research Council of Turkey (TUBITAK). His research interests include security for cyber-physical systems, Internet of Things, security for critical infrastructure networks, IoT security, security analytics, social networks, big data analytics, distributed computing, wireless networks, wireless ad hoc and sensor networks, localization, and P2P networks.

A. Selcuk Uluagac (Member, IEEE) received the MSc degree in Electrical and Computer Engineering (ECE) from Carnegie Mellon University, Pittsburgh, Pennsylvania, the MSc degree in information security from the School of Computer Science, Georgia Tech, Atlanta, Georgia, in 2002 and 2009, respectively, and the PhD degree with a concentration in information security and networking from the School of Electrical and Computer Engineering (ECE), Georgia Tech, Atlanta, Georgia, in 2010. He is currently an associate professor with the Department of Electrical and Computer Engineering, Florida International University (FIU), Miami, Florida. Before joining FIU, he was a senior research engineer with the School of Electrical and Computer Engineering (ECE), Georgia Institute of Technology, Atlanta, Georgia. Prior to Georgia Tech and senior research engineer at Symantec. The focus of his research is on cybersecurity topics with an emphasis on its practical and applied aspects. He is interested in and currently working on problems pertinent to the security of cyber-physical systems and Internet of Things. In 2015, he received a Faculty Early Career Development (CAREER) Award from the US National Science Foundation (NSF). In 2015, he was awarded the US Air Force Office of Sponsored Research (AFOSR)'s 2015 Summer Faculty Fellowship. He is also an active member ACM, and ASEE and a regular contributor to national panels and leading journals and conferences in the field. He has served on the program committees of top-tier security conferences such as IEEE S&P Oakland, NDSS, ACM ASIACCS, inter alia. He was the general chair of ACM Conference on Security and Privacy in Wireless and Mobile Networks (ACM WiSec), in 2019. Currently, he is the area editor of the *Elsevier Computer Networks Journal* and serves on the editorial board of the *IEEE Communication Surveys and Tutorials*. For more information, please visit http://web.eng.fiu.edu/selcuk.

Kemal Akkaya (Senior Member, IEEE) received the PhD degree in computer science from the University of Maryland Baltimore County, Baltimore, Maryland, in 2005 and joined the Department of Computer Science, Southern Illinois University (SIU), Carbondale, Illinois as an assistant professor. He is a full professor with the Department of Electrical and Computer Engineering, Florida International University, Miami, Florida. He was an associate professor at SIU from 2011 to 2014. He was also a visiting professor at The George Washington University, Washington, D.C. in Fall 2013. He leads the Advanced Wireless and Security Lab (ADWISE) in the Electrical and Computer Engineering (ECE) Department. His current research interests include security and privacy, security and privacy, Internet-of-Things, and cyber-physical systems. He is the area editor of the *Elsevier Ad Hoc Network Journal* and serves on the editorial board of the *IEEE Communication Surveys and Tutorials*. He was the general chair of IEEE LCN 2018 and TPC chair for the *IEEE ICC Smart Grid Communications*. He has served as the guest editor for many journals and in the OC/TPC of many leading network/security conferences including IEEE ICC, Globecom, INFOCOM, LCN and WCNC, and ACM WiSec. He has published more than 180 papers in peer-reviewed journal and conferences. He has received "Top Cited" Article Award from *Elsevier*, in 2010.