AuthoRing: Wearable User-presence Authentication

Xiaohui Liang
University of Massachusetts Boston
Boston, Massachusetts, 02125

David Kotz
Dartmouth College
Hanover, New Hampshire, 03755

ABSTRACT

A common log-in process at computers involves the entry of username and password; log out depends on the user to remember to log out, or a timeout to expire the user session. Once logged in, user sessions may be vulnerable to imposter attacks in which an impostor steps up to the user's unattended computer and inherits the user's access privilege. We propose a ring-based authentication system called "AuthoRing", which restricts the imposter attackers from generating new inputs at the computer's mouse and keyboard. During the log-in process, an eligible AuthoRing user wears a digital ring with accelerometers and wireless communication capability. When input is detected at the mouse or keyboard, the computer's AuthoRing system correlates hand-motion data received from the ring with the input data from the computer's window manager, and detects imposter attacks when these data are insufficiently correlated. We implemented the AuthoRing system and evaluated its security, efficiency, and usability; we found that imposter attacks can be effectively detected and the required operations happen quickly with negligible delays experienced by the user.

1. INTRODUCTION

Computers, mobile devices, and applications need to verify that their user is who she says she is before taking any actions that expose sensitive data or trigger sensitive transactions. For example, electronic health record apps present a patient's health information to authorized doctors and nurses and banking apps reveal credit-card transactions to credit-card owners. After a user is authenticated – typically with a password or PIN code – the device tracks the user's activity; if an inactivity period exceeds a timeout threshold, the user is automatically logged out (de-authenticated).

Although common, this type of authentication (and de-authentication) has several security and usability problems. Consider a user visits a public computer in a library to write files under her account at the computer. While the computer is a shared device, if the user forgets to logout, someone could later try to access the files created by the previous user. Another case is a personal computer left unattended in a public space; an *impostor* (perhaps a fellow employee, or a visitor with no authority to use the system) may physically gain access to the unattended user session, with access to a victim's account. The *impostor attacks* are particularly difficult to detect and trace because all actions were apparently taken by the authorized user, and the victim account was not associated with suspicious activity before [8, 12].

One way to limit the imposter attacks is to set a small timeout, and require the user to input her password to re-authenticate whenever she has been inactive. On the other hand, users find it inconvenient to enter their password so frequently. Moreover, recent research has demonstrated that entering passwords in a public place may not be secure; wireless signals and cameras may be used to eavesdrop on keystrokes [4, 16]. How can a user remain authenticated with minimum effort and without re-entering her password – and yet prevent access by imposters that step in to use her unattended computer? We aim to achieve this goal.

We propose a continuous user-presence authentication system, called AuthoRing, in the form of a ring that assists a user in continuously confirming her presence through computer mouse motion. AuthoRing aims to authenticate the user who is actually using the computer, while existing proximity-based schemes [10, 7, 11] authenticate the user who is physically close to the computer. AuthoRing requires no changes to the computer hardware and operating system. The user wears a digital ring on her finger; this ring is equipped with an accelerometer and Bluetooth Low Energy (BLE), technology already demonstrated for use in gesture recognition [14, 9]. AuthoRing makes three contributions:

- Addressing an important security risk by verifying user presence and preventing impostors from hijacking unattended user sessions. AuthoRing verifies user presence by correlating mouse motion with hand-motion data. The verification can be accomplished by a user either passively or actively, while achieving high accuracy as shown in our experiments.
- Improving usability (and reducing vulnerability) by avoiding repeated instances of password entry. AuthoRing collects mouse-motion and hand-motion data passively and largely reduces the need for active authentication. For an active authentication, the user can simply wiggle the mouse to verify her presence.
- Implementing and evaluating the AuthoRing system. We implement the AuthoRing system as Python programs

WearSys '17, June 19, 2017, Niagara Falls, NY, USA.

DOI: http://dx.doi.org/10.1145/3089351.3089357

and build a ring prototype with TinyDuino boards. We show that with appropriate system parameters, passive authentication can be accomplished by a two-second mouse motion while the false-reject ratio was close to 0 and the false-accept ratio was below 10%. For a four-second mouse motion, the false-accept ratio was reduced to 4%.

2. DESIGN GOALS

We consider a scenario that involves an AuthoRing-enabled computer, and a user wearing a ring with embedded accelerometer. The ring and the computer both have Bluetooth Low Energy (BLE), and have successfully paired. AuthoRing has two goals.

Resisting impostor attacks. An impostor attacker aims to impersonate a victim user and gain an access to her account. (In the literature, such attacks are sometimes called "lunchtime attacks," in which an adversary temporarily gains access to a co-worker's workstation while the co-worker is away.) The challenge here is to enable the computer to recognize the right user without imposing undue burden on the user or adding any noticeable delay in computer interactions. Note that, in some scenarios, the adversary may be an authorized user of the computer – but is seeking unauthorized access to another user's unattended session. Furthermore, we want to prevent impostor access even when the authentic user is nearby (within BLE range, perhaps at a nearby lunch table) but not actively using the computer.

Minimizing the use of password. Repeatedly entering passwords might be annoying and recent research shows the potential risk of leaking credentials to eavesdropping attackers [4, 16]. When the authentic user wears an AuthoRing, it can assist the user in re-authentication. She just needs to prove to the wearable and the computer that she is present at the moment. Means to develop an accurate and usable user-presence authentication is the key to minimizing the use of traditional password-based user authentication. AuthoRing takes a key step in this direction.

3. THREAT MODEL

We assume the impostor attacker is aware of the AuthoRing system and aims to break the user-presence authentication. We consider two types of impostor attackers: (1) the attacker randomly moves the mouse; and (2) the attacker observes the user's hand motion and moves the mouse to mimic the user's hand motion. We assume the attacker neither disrupts the AuthoRing system nor installs any malicious software on the computer. Although savvy attackers may use such tools, we are motivated by the opportunistic "lunchtime" attacker. Any attacker involving such attack tools requires a different range of solutions, including anti-malware tools and stronger authentication when software is installed, de-installed, or modified. Such tools should certainly be used in addition to AuthoRing, and are well-established products.

4. AUTHORING

The AuthoRing system includes the following components.

4.1 Initialization

When a user first uses a computer, she connects her ring to the computer via BLE. AuthoRing adopts a pairing protocol for small embedded devices that have no display but accelerometers; there are many approaches and specific methods are beyond the scope of this paper. After being connected, the computer and the ring communicate via a secure channel as long as they are within BLE communication range.

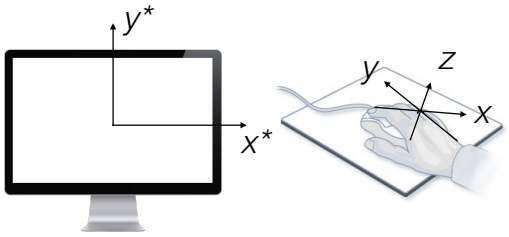


Figure 1: Five axes

4.2 User-presence authentication

Our user-presence authentication needs to collect two kinds of motion information: the AuthoRing system on the computer collects the mouse motion while the ring collects the hand motion. When the AuthoRing system detects mouse motion on the computer, it starts to save the mouse-motion data and it notifies the ring so the ring begins to collect acceleration data about the user's hand motion. When the mouse is stationary or the collected data is enough, the computer notifies the ring, which stops the data collection and forwards the data to the computer.

We envision a three-axis accelerometer, and the mouse moves across a two-dimensional screen; thus, we work with five axes as in Figure 1. Denote the mouse-motion data $M = \{m_1, \cdots, m_k\}$ at time points $\{t_{m,1}, \cdots, t_{m,k}\}$ and the hand-motion data $N = \{n_1, \cdots, n_h\}$ at time points $\{t_{n,1}, \cdots, t_{n,h}\}$. Since the computer and the ring collect the data almost from the same period, we have $t_{m,k} \approx t_{n,h}$, $t_{m,1} \approx t_{n,1}$. The more data they collect, the better accuracy the authentication should achieve. However, to accumulate a longer period of data for authentication compromises usability (forcing the user to engage more). Here, we define a threshold $\mathcal{T}$ as a threshold to trigger the termination of data collection; if the data period is longer than $\mathcal{T}$, i.e., $t_{m,k} - t_{m,1} (\approx t_{n,h} - t_{n,1}) \geq \mathcal{T}$, the data collection period ends and the authentication moves forward to the correlation algorithm. Denote $t_{min} = \min\{t_{m,1}, t_{n,1}\}$ and $t_{max} = \max\{t_{m,k}, t_{n,h}\}$.

Correlating the mouse coordinates with the acceleration data involves three challenges: i) the two sets of data are in different units, and it is difficult to transform one unit to the other; ii) the orientation of the accelerometer on the ring is always changing because the hand and the mouse are in motion; and iii) the changes of mouse coordinates are difficult to compare to accelerations because of the different sampling frequency. From our preliminary results, we found axis alignment of the data using gravity was computationally expensive and unreliable, so we do not align the axes (x^*, y^*) with any of (x, y, z). We observed that a change of coordinate in x^* or y^* could result in a change of acceleration data in all three axes (x, y, z). Thus, we explore the pairwise correlation of six pairs of data where $(c_{x^*,i}, c_{y^*,i})$ are two screen coordinates and $(a_{x,j}, a_{y,j}, a_{z,j})$ are three ac-

celeration readings:

$$(c_{x^*,i}, a_{x,j}), (c_{y^*,i}, a_{x,j}), (c_{x^*,i}, a_{y,j}),$$
$$(c_{y^*,i}, a_{y,j}), (c_{x^*,i}, a_{z,j}), (c_{y^*,i}, a_{z,j}).$$

The correlation algorithm includes three steps: peak detection, weight calculation, and distance calculation.

Peak detection. We find the peaks in the time series for each of five axes (x^*, y^*, x, y, z). The idea here is that changes of direction of mouse motion are where the peak accelerations also occur, so finding peaks in mouse-motion coordinates is essentially finding the places where the mouse changed direction, and hence should occur at the same time as peak accelerations. We used a low-pass filter to ignore small peaks in the acceleration data.

Weight calculation. The time-series data may contain noise due to small movements of the mouse and the hand. As such, if a given time series contains distinctive features, it should be given a larger weight in the correlation. Here, the distinctive features are defined by the number of peaks and the interval between these peaks. We assign a relatively larger weight to the time series that has moderate peaks with varying time intervals. On the other hand, if the peaks are too many or too few, the weight is relatively low. We specifically discuss the weight calculation in the evaluation section. We denote five weights respectively for five axes $\{w_{x^*}, w_{y^*}, w_x, w_y, w_z\}$. We calculate normalized weights for all six pairs:

$$w_{\alpha,\beta} = \frac{w_\alpha w_\beta}{(w_{x^*} + w_{y^*})(w_x + w_y + w_z)}$$

where $\alpha \in \{x^*, y^*\}$ and $\beta \in \{x, y, z\}$.

Distance calculation. To calculate the distance, we use the number of peaks and the time interval between these peaks. After extracting the peaks from the time-series data, we use these peaks to generate a function F as follows. For $\alpha \in \{x^*, y^*\}$ and $\beta \in \{x, y, z\}$, we define, for $t = [t_{min}, t_{max}]$,

$$F_\alpha(t) = \begin{cases} 1 & \text{if a peak is in } [t - \Delta t, t + \Delta t] \text{ of axis } \alpha; \\ 0 & \text{otherwise.} \end{cases}$$

$$F_\beta(t) = \begin{cases} 1 & \text{if a peak is in } [t - \Delta t, t + \Delta t] \text{ of axis } \beta; \\ 0 & \text{otherwise.} \end{cases}$$

where Δt is a parameter set to tolerate the time delay of peaks in two different time-series data caused by computation, communication, and hardware sensing. Denote $F_{\alpha,\beta}(t) = F_\alpha(t) | F_\beta(t)$ to merge the non-zero periods (NZPs), i.e., if either $F_\alpha(t)$ or $F_\beta(t)$ equals 1, $F_{\alpha,\beta}(t)$ equals 1. A distance function for axes α and β is:

$$D_{\alpha,\beta} = 2 \times \frac{\text{the number of NZPs in } F_{\alpha,\beta}}{\text{the number of NZPs in } F_\alpha \text{ plus that in } F_\beta} - 1$$

If F_α and F_β perfectly match, $D_{\alpha,\beta}$ should be the minimum $2 \times \frac{1}{2} - 1 = 0$; otherwise, the number of NZPs in $F_{\alpha,\beta}(t)$ is larger and the $D_{\alpha,\beta}$ is larger. Finally, we define the correlation score $\eta_{M,N}$ as follows.

• If there exists at least one pair of $\alpha \in \{x^*, y^*\}$ and $\beta \in \{x, y, z\}$ such that $w_\alpha \geq \tau$ and $w_\beta \geq \tau$, where τ is an adjustable threshold, the two time series represented by α and β are considered to have comparable distinctive features. The correlation score is defined as

$$\eta_{M,N} = min\{D_{\alpha,\beta} \mid w_\alpha \geq \tau \text{ and } w_\beta \geq \tau \text{ for } \alpha, \beta\}.$$

• If there exists no such pair, we calculate the weighted average distance of all six pairs as the correlation score.

$$\eta_{M,N} = \sum_{\alpha=(x^*,y^*),\beta=(x,y,z)} w_{\alpha,\beta} \times D_{\alpha,\beta}$$

The minimum score of $\eta_{M,N}$ is 0. The authentication succeeds if $\eta_{M,N}$ is below a pre-defined threshold; fails otherwise. Details are shown in the evaluation section.

4.3 Optimized authentication

Our user-presence authentication can be unobtrusive and efficient if the computer mouse is moved by the user frequently because there is plentiful data to be used by the correlation algorithm. In case the user does not frequently move the mouse, AuthoRing takes the following approaches.

Mouse wiggling when needed. The AuthoRing system accumulates the mouse and hand-motion data in a passive manner and can thus verify user presence immediately when it is really needed. Our correlation algorithm could be run over the passively accumulated mouse motion. When a mouse click and a keyboard input is detected, user presence is immediately verified. In this case, the user does not need to actively wiggle the mouse. However, if a mouse click or a keyboard input is detected and there is insufficient recent data collected passively, the user needs to actively wiggle the mouse to generate more data for authentication. Wiggling the mouse is an intuitive and convenient approach, as users are accustomed to wiggling a mouse to wake the computer, and this action is easier than re-entering a password.

Individual interaction period. We assume that the impostor attacker is unable to successfully step into a user session within a short time after the previous user departs, without being detected. We define a term *Individual Interaction Period* (IIP) as a time period during which a computer receives a sequence of user inputs, such as keystrokes, mouse motion, mouse clicks, and mouse scrolling, and the inactive gap between any two consecutive inputs is $\leq T_{IIP}$. T_{IIP} is an adjustable parameter depending on the security requirement and the physical environment. In other words, the AuthoRing system considers two user inputs with an inactive gap $\leq T_{IIP}$ to be from the same user. The AuthoRing system maintains one IIP at a time. A new IIP is created when a user input is received and the inactive gap between this input and the last input is $> T_{IIP}$. For a given IIP, only one authentication (either passive or active) is needed. This strategy largely reduces the need for the user-presence authentication when the mouse clicks and keyboard inputs are frequent.

5. EVALUATION

In our evaluation, we used a desktop computer that is equipped with a mouse and an IOGEAR Bluetooth 4.0 USB Micro Adapter. We implemented a ring prototype with TinyDuino, an open-source platform that features the full capabilities of the popular Arduino platform but miniaturizes the board [2]. We did not record the content of their activity, just the time and type of interactions. All human-subject experiments in this paper were approved by our Institutional Review Board (IRB).

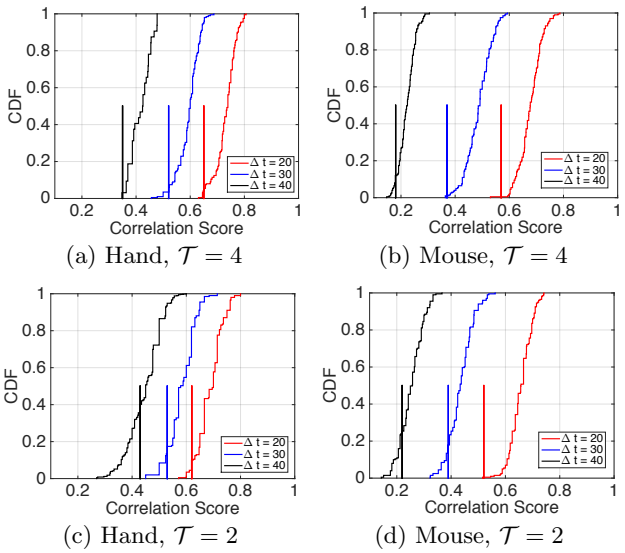
5.1 Evaluation of Correlation

We asked five users to wear our ring prototype and use our desktop computers. These users reported that their in-

teraction behavior changed at first, but after they found a comfortable way to use the mouse and keyboards, wearing our prototype had negligible impacts on their interaction behavior. During the data collection process, we set the mouse tracking sensitivity at moderate and fixed level. For each user, AuthoRing collected both mouse-motion data and hand-motion data simultaneously for about 10-15 minutes. The sampling frequencies of mouse/hand motion data both were set to 50 Hz, so for each user we obtained around 30,000 – 45,000 data points of mouse motion and hand motion. If their mouse usage filled about 30 – 50% of the total time, we obtained about 9,000 – 22,500 effective data points. From the effective data points we examined small windows of $\mathcal{T}$-length by shifting a sliding window one data point at a time to evaluate the correlation over the entire data range where $\mathcal{T}$ is the minimum time period of accumulative mouse-motion data to trigger the user-presence authentication. Here we have two options: one is to randomly choose one data window from the hand-motion data and correlate it with all windows from the mouse-motion data; and the other is to randomly choose one mouse-motion window and correlate it with all windows from the hand-motion data. Ideally, the correlation should succeed if and only if the two windows correspond to the same time period.

After choosing two windows, we first used a peak-detection algorithm to derive the peaks in each window, and calculated a weight based on the number of peaks. We set a small weight for a window if it had too many or too few peaks, and a large weight for a window if it has a moderate number of peaks. Specifically, we use a log-normal function $w = \text{lognpdf}(num/40, \mu, \sigma)$ to determine the relation between the number of peaks num and the weights w. The log-normal function aligns with our requirements on setting the weights where a convex function has $\text{lognpdf}(0) = 0$ and $\text{lognpdf}(+\infty) \to 0$. In the following, we describe experiments for $\mu = 0$ and $\sigma = 0.4$. Additionally, in our correlation algorithm, we chose a tolerance range Δt to handle the time delay offset of peaks appearing in the mouse-motion data and the hand-motion data; such delay was due to the transmission time of the notifications from the computer to the ring. We chose $\Delta t = 20, 30, 40$ ms. Intuitively, the more data the correlation algorithm used, the better accuracy it could achieve. Here, we chose $\mathcal{T} = 2$ s or 4 s to control the size of the input data.

In Figures 2(a) and (b), we plot the cumulative distribution function (CDF) of correlation scores for $\mathcal{T} = 4$ s. In Figures 2(c) and (d), we plot the CDF for $\mathcal{T} = 2$ s. In both Figures 2(a) and (c), we report the result of a randomly-selected window from the hand-motion data and all windows from the mouse-motion data. In Figures 2(b) and (d), we report the result of a randomly-selected window from the mouse motion and all windows from the hand-motion data. Note that the correlation results for other selections are similar because we scan the all windows of the motion data. In each of these figures, three CDF are generated for $\Delta t = 20, 30, 40$ ms, respectively from right to left. We also plot the correlation score, as a vertical line, of the two windows that occurred at the same time (i.e., the two windows that should correlate best). If we choose a threshold larger than that score, we obtain a false-reject ratio close to 0 and we obtain the corresponding false-accept ratio at the intersection of the CDF and the vertical line. In Figure 2(b), comparing $\Delta t = 30$ and 40 ms, we found the false-accept



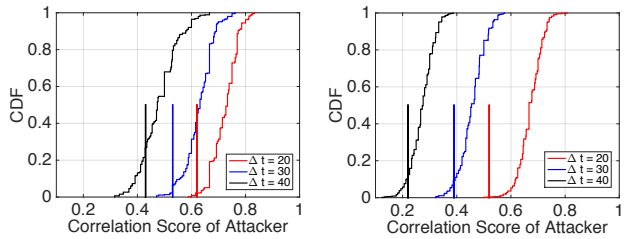
(a) Hand, $\mathcal{T} = 4$ (b) Mouse, $\mathcal{T} = 4$

(c) Hand, $\mathcal{T} = 2$ (d) Mouse, $\mathcal{T} = 2$

Figure 2: CDF of correlation scores

ratio was not always smaller if we choose a larger Δt. A larger Δt enables the correlation algorithm to tolerate more delay caused by the communication, computation, and hardware sensing, but it also allows the correlation algorithm to falsely accept other windows (which did not occur in the correct time window). From Figure 2(a) and (b) where $\mathcal{T} = 4$ s, we found that when $\Delta t = 20$ ms, the false-accept ratio was about 4% while the false-reject ratio was close to 0; when $\Delta t = 30, 40$ ms, the false-accept ratio was still below 10%. From Figures 2(c) and (d) where $\mathcal{T} = 2$ s, it can be seen that when $\mathcal{T}$ decreased from 4 s to 2 s, the false-accept ratio generally increased (while keeping the false-reject ratio close to 0). The main reason is because the size of each window is smaller, each window could more likely be falsely correlated with others. Specifically, we had large false-accept ratios 40% and 25% when $\mathcal{T} = 2$ s and $\Delta t = 40$ ms. To reduce the false-accept ratio, we choose $\Delta t = 20$ ms instead of 40 ms, for which the false-accept ratios then decreased to below 10%. In other words, we have less tolerance on the time delay, and that in turn impose a more restricted requirement on the sensing response and the transmission speed of the sensor data. In sum, these findings suggest we either choose $\mathcal{T} = 4$ s, or $\mathcal{T} = 2$ s with a small Δt.

We then asked the users to actively wiggle the mouse after they saw a notification until the notification disappears. The notification appeared for 10 seconds. We observed all five users wiggle the mouse faster and stronger than they normally move the mouse partly because they do not need to purposely move the mouse to a specific location. In this case, the corresponding mouse motion and hand motion data contained more peaks with shorter time intervals. We applied a similar analysis method and found better correlation results (false-reject ratio close to 0 and false-accept ratio below 1% even for two-second motion) than those from the passive experiment. Although the authentication accuracy can achieve 1% for active authentication and 4% for passive authentication, the data used for matching was drawn randomly. In other words, if the impostor attacker randomly generated either the mouse-motion data or the hand motion data, it can be detected by the user-presence authentication with a large probability. However, in practice, the impostor



(a) Hand motion, 2s (b) Mouse motion, 2s

Figure 3: CDF of correlation score by attacker

attacker could generate better motion data by observing the user's behavior. In the following, we evaluate the capability of such an intelligent attacker and analyze the security of AuthoRing.

5.2 Evaluation of Security

We considered different ways to simulate an impostor attacker. We decided to let users be their own attacker because we expect that they may mimic their own behavior better than other people. Specifically, we videotape the users' mouse usage in previous experiments, and additionally videotape the users' hand motion when they are not using the mouse. After two days we contact the user again to continue our experiment. We delay two days because we want the user to forget any particular gesture they used in the first experiment (because the attacker will not know these gestures in advance). This time, we show them a two-second video segment of either mouse (cursor) motion or hand motion of the videos. We want the user to repeat the mouse motion by observing the hand motion, or vice versa. We require the user to mimic the motion in the video in real-time and the data collection starts and ends as the video starts and ends (because the computer and the ring start and end the data collection at the same time). We randomly selected a two-second video segment, so the user (attacker) does not have time to prepare her gesture, although she may become more familiar with the gestures after multiple performances. We plot the CDF of correlation scores in Figure 3(a) where the user watched a video segment of hand motion, and Figure 3(b) where the user watched a video segment of mouse motion. We found that such an impostor attacker did not do better than the attacker who guessed randomly. Indeed, our authentication had a *lower* false-accept ratio for this mimicking attack. An examination of the data found the main reason for this result is the natural delay in human response. As indicated by a study from 17,000,000 tests with live subjects [1], the most common human reaction time is around 265 ms, and the fastest is about 160 ms. The average delay shown from that data is about 200 ms. In our algorithm, we set $\Delta t = 20, 30, 40$ ms, which results in an allowable time range of 40, 60, or 80 ms around an event, which is far less than the fastest human reaction time. Therefore, the impostor's malicious behavior unavoidably incurs a delay due to the attacker's limited reaction time, while our correlation algorithm does not tolerate such delay. As such, the impostor is unable to successfully attack the AuthoRing system by mimicking the user when the user is visible but away from her computer. We understand that a more sophisticated attacker could use cameras and robotics to mimic the victim's behavior, but that approach will certainly raise the bar of complexity for the attacker!

6. DISCUSSION

In this section, we discuss a few issues and alternatives about security, scenarios, and techniques.

• **Security:** *Ring loss or theft.* A wearable ring may be lost or stolen by an attacker. AuthoRing could integrate some existing techniques to prevent the attacker from using the ring to impersonate the user. For example, Apple Watch automatically locks itself if it detects no skin contact, and requires a user to enter a passcode code to unlock the watch. Similarly in AuthoRing, the wearable ring should lock itself if it detects no skin contact. To unlock the ring, a combination of 3D gestures could be preset by a user and later used as a secret to verify the user [5]. Another interesting solution could be using bioimpedance measurement to develop a biometric around a user's ring finger [6].

Comparison with proximity-based solutions. Proximity-based schemes [10, 7, 11] authenticate users if they are in the proximity of the device as determined by the wireless radio signal strength from a token they carry. The Bluetooth and NFC techniques can be used to implement proximity-based solutions. These schemes are better than AuthoRing because they work even for users who are not using the computer mouse with the ring hand. Although these schemes provide continuous authentication, they may not authenticate the user who is actually using the device. AuthoRing correlates the mouse motion and hand motion and is able to continuously determine if it is the same user that uses the computer.

• **User scenarios:** *Multiple-user scenario.* For a public computer that could be shared by multiple users, if multiple users are in the proximity of the computer and their rings are all connected to the computer, the computer is unable to know which user's hand motion should be collected for authentication. The rings cannot tell if the user is using the computer or not. As such, AuthoRing needs the user to select her ring (e.g., in a pop-up window) when the user switches to a new computer.

Multiple-user ring. In AuthoRing, the ring need not be a personal device. Consider a workplace at which the user picks up a random ring at the entry desk, uses the ring for a whole day, and returns the ring back to the service desk after work. A user needs to know the ring name and select the ring name as above, when beginning work on a computer. As such, the ring can be used by different users in another day, because it does not store any personal secret.

• **Improved techniques:** *Extending AuthoRing to use keyboard input and hand motion.* AuthoRing could use the keystrokes to further improve its confidence about user's presence. For example, if the detected keystrokes and the hand motion are highly correlated while the mouse is stationary, the AuthoRing system knows the user is using the keyboard. However, when they are not correlated, the AuthoRing system has no idea about the current user, and cannot reject a user based on the keystroke information.

Mobile devices. AuthoRing might be extended to smartphones and tablets, and to other input interfaces such as touch screen and touch pad. For these interfaces, a user typically uses one finger; for AuthoRing to work, the user has to wear the ring on the primary finger. However, such motion may be very slight, and the corresponding correlation algorithm has to be improved to accurately match the two kinds of motion information. More study is needed to extend AuthoRing in this way.

Implement AuthoRing in a smartwatch. The idea behind AuthoRing could perhaps be similarly implemented in a smartwatch. However, we found when users interact with the mouse and keyboards, the wrist motion was much less intensive than the hand motion. As such, the correlation algorithm needs a much longer period of sensor data to verify the user's presence and guarantee the desired security properties. In this case, an active user-presence authentication requiring 10 s active effort is even worse than a password entry in terms of usability.

7. RELATED WORK

Continuous authentication aims to resist impostor attacks by implicitly and continuously performing user authentication without disrupting the normal user-device interaction. Pico [10] detects whether the user and the Pico token are in the communication range of the computer, but fails when the user is present but not the current user of the computer. Continuous authentication could also be implemented with a behavioral biometric, such as eye movement [8], keystroke patterns [3], mouse dynamics [13], or via correlation with wrist motion as in ZEBRA [12]. These other solutions introduce errors and delays. For example, in ZEBRA [12], even after the attacker takes control of the computer for more than 5 seconds, the attack detection probability is less than 80%. In the eye-movement detection system [8], to achieve a false-reject ratio of 0, the system has a false-accept ratio of 19.2%. The mouse-dynamic system [15] achieves a false-accept ratio of 8.74% and a false-reject ratio of 7.69% with a corresponding authentication time of 11.8 seconds. As such, they all are ineffective to resist quick impostor attacks as they need a relatively long time to receive enough behavioral information for attack detection. In comparison, AuthoRing is more efficient and accurate than previous solutions, and it further balances the usability and the security with a passive and an active user-presence authentication.

8. CONCLUSIONS

In this paper, we introduce AuthoRing, a continuous user-presence authentication system that assists users in securely and efficiently confirming their presence through mouse motion. AuthoRing software detects when the computer is used by an imposter (anyone other than the ring wearer) and blocks access to the logged-in user sessions. We implemented the AuthoRing system in the computer using Python and built a ring prototype using TinyDuino. Through our experiments conducted in a lab setting, we show that if the motion data was collected passively, the false-reject ratio was close to 0 while the false-accept ratios were 10% and 4% respectively for two-second and four-second motion samples. If the user actively wiggles the mouse to generate the motion data, the false-accept ratios were further reduced to 1% for two-second motion samples. We found that a sophisticated impostor, who mimics mouse (or hand) motion by observation, failed due to the inherent limitations of the human reaction delay. Given this proof of concept, our next step is to deploy a pilot study and conduct more comprehensive experiments to evaluate AuthoRing in realistic environments.

9. ACKNOWLEDGEMENTS

This research results from a research program at the Institute for Security, Technology, and Society at Dartmouth College, supported by the National Science Foundation under award number CNS-1329686. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the sponsors.

10. REFERENCES

[1] Human Bench Mark. http://www.humanbenchmark.com/tests/reactiontime.

[2] Tinyduino. https://tiny-circuits.com/.

[3] Banerjee, S. P., and Woodard, D. L. Biometric authentication and identification using keystroke dynamics: A survey. *Journal of Pattern Recognition Research 7*, 1 (2012), 116–139.

[4] Chen, B., Yenamandra, V., and Srinivasan, K. Tracking keystrokes using wireless signals. In *MobiSys* (2015), pp. 31–44.

[5] Chong, M. K., Marsden, G., and Gellersen, H. GesturePIN: using discrete gestures for associating mobile devices. In *Proceedings of the International Conference on Human Computer Interaction with Mobile Devices and Services (Mobile HCI)* (2010), pp. 261–264.

[6] Cornelius, C., Peterson, R., Skinner, J., Halter, R., and Kotz, D. A wearable system that knows who wears it. In *MobiSys* (2014), pp. 55–67.

[7] Corner, M. D., and Noble, B. D. Protecting applications with transient authentication. In *MobiSys* (2003), pp. 57–70.

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

[9] Gummeson, J., Priyantha, B., and Liu, J. An energy harvesting wearable ring platform for gesture input on surfaces. In *MobiSys* (2014), pp. 162–175.

[10] Hermans, J., and Peeters, R. Realizing Pico: Finally no more passwords! *IACR Cryptology ePrint Archive* (2014), 519.

[11] Landwehr, C. E. Protecting unattended computers without software. In *ACSAC* (1997), pp. 274–283.

[12] Mare, S., Molina-Markham, A., Cornelius, C., Peterson, R., and Kotz, D. ZEBRA: Zero-effort bilateral recurring authentication. In *IEEE S&P* (2014), pp. 705–720.

[13] Nakkabi, Y., Traoré, I., and Ahmed, A. A. E. Improving mouse dynamics biometric performance using variance reduction via extractors with separate features. *IEEE Transactions on Systems, Man, and Cybernetics, Part A 40*, 6 (2010), 1345–1353.

[14] Nirjon, S., Gummeson, J., Gelb, D., and Kim, K.-H. TypingRing: A wearable ring platform for text input. In *MobiSys* (2015), pp. 227–239.

[15] Shen, C., Cai, Z., Guan, X., Du, Y., and Maxion, R. User authentication through mouse dynamics. *IEEE Transactions on Information Forensics and Security 8*, 1 (Jan 2013), 16–30.

[16] Shukla, D., Kumar, R., Serwadda, A., and Phoha, V. V. Beware, your hands reveal your secrets! In *ACM CCS* (2014), pp. 904–917.