Press @$@$ to Login: Strong Wearable Second Factor Authentication via Short Memorywise Effortless Typing Gestures

Prakash Shrestha* — Equifax Inc. — prakash.shrestha@equifax.com
Nitesh Saxena — University of Alabama at Birmingham — saxena@uab.edu
Diksha Shukla† — University of Wyoming — dshukla@uwyo.edu
Vir V. Phoha — Syracuse University — vvphoha@syr.edu

2021 IEEE European Symposium on Security and Privacy (EuroS&P) | 978-1-6654-1491-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/EUROSP51992.2021.00016

***Abstract*—The use of wearable devices (e.g., smartwatches) in two factor authentication (2FA) is fast emerging, as wearables promise better usability compared to smartphones. Still, the current deployments of wearable 2FA have significant usability and security issues. Specifically, one-time PIN-based wearable 2FA (PIN-2FA) requires noticeable user effort to open the app and copy random PINs from the wearable to the login terminal's (desktop/laptop) browser. An alternative approach, based on one-tap approvals via push notifications (Tap-2FA), relies upon user decision making to thwart attacks and is prone to skip-through. Both approaches are also vulnerable to traditional phishing attacks.**

To address this security-usability tension, we introduce a fundamentally different design of wearable 2FA, called *SG-2FA*, involving wrist-movement *"seamless gestures"* captured *near transparently* by the second factor wearable device *while* the user types a *very short* special sequence on the browser during the login process. The typing of the special sequence creates a wrist gesture that when identified correctly uniquely associates the login attempt with the device's owner. The special sequence can be fixed (e.g., "@$@$"), does not need to be a secret, and does not need to be memorized (could be simply displayed on the browser). This design improves usability over PIN-2FA since only this short sequence has to be typed as part of the login process (no interaction with or diversion of attention to the wearable and copying of random PINs is needed). It also greatly improves security compared to Tap-2FA since the attacker can not succeed in login *unless* the user's wrist is undergoing the exact same gesture at the exact same time. Moreover, the approach is *phishing-resistant* and *privacy-preserving* (unlike behavioral biometrics). Our results show that SG-2FA incurs only minimal errors in both benign and adversarial settings based on appropriate parameterizations.

I. Introduction

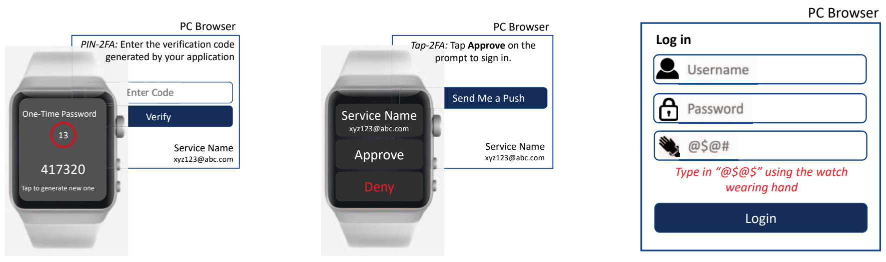
Mobile app-based two-factor authentication (2FA) is widely deployed on the Internet today. This approach works by verifying something the user knows, i.e., a password, and something the user possesses i.e., a general-purpose device, traditionally a smartphone, running the app. In contrast with hardware token based 2FA (e.g., YubiKey[1], RSA SecureID [32], and FIDO U2F [50]), the use of app-based 2FA helps improve usability and deployability of 2FA. A commonly used app-based 2FA scheme, such as Google 2SV [17] and Celestix's HOTPin [6], requires the user to enter his password and copy a random and one-time PIN (OTP) from the mobile app over to the web browser of the authentication terminal (laptop/desktop). One-tap approach for two-factor authentication (Tap-2FA), such as Duo Push [11] and Google Prompt [18], is another variant of app-based 2FA that helps improve the usability of 2FA as the user simply needs to tap on a login attempt push notification shown on the app's interface.

Rapid Emergence of Wearable 2FA: Beyond smartphones, mobile apps of these 2FA variants are being rapidly deployed and gaining momentum on wrist-worn wearables, referred to as *Watch-2FA* (e.g., Google Auth [19], SAASPASS [34] for PIN-2FA, and Google Prompt [36], Duo Push [10] for Tap-2FA variant). Sample snapshots of these deployed Watch-2FA schemes are presented in Figure 1a and 1b. Wrist-wearables, especially the smartwatches, are a compelling platform to implement 2FA since these devices are gaining popularity in the consumer space [14], and they make 2FA schemes easier for the user compared to the smartphone. Unlike the phone, since wrist-wearables may always remain attached to the user, the user does not need to explicitly look for/reach out to the watch, thereby making the use of 2FA schemes relatively easier for the user compared to the 2FA schemes used with the phone.

Fundamental Problems with Current Deployments: Unfortunately, the aforementioned Watch-2FA schemes still either require significant user-effort (PIN-2FA) or are prone to user errors, negligence or click-through (Tap-2FA). PIN-2FA requires the user to interact with the watch, thereby diverting the user's attention away from the authentication terminal. Specifically, PIN-2FA requires the user to launch the app installed on the watch, read and copy the OTP to the authentication terminal's browser. The small-form factor of watches may make it difficult for the user to launch the app and read the OTP, and hence negatively impact the usability of 2FA. The reduced usability may discourage users from adopting this approach in practice, similar to phone-based PIN-2FA [12], [8], [22].

While Tap-2FA makes the process easier for the user, it still involves user-watch interaction More critically, in Tap-2FA, the user is supposed to read and verify the login information shown on the login prompt (e.g., service name, account name,

*Work done at UAB †Work done at SU



(a) PIN-2FA (b) Tap-2FA (c) SG-2FA

Fig. 1: Depiction of the watch UIs of two deployed variants of Watch-2FA and SG-2FA. In our proposed SG-2FA scheme, the watch UI is irrelevant as the scheme requires zero interaction with the watch, and only requires the user to type a short, fixed sequence of characters on the browser of the terminal.

location, etc.). Unlike the phone, the small-form factor of the watch may make it difficult for the user to view and read the crucial login information. Given this, the user is likely to accept or skip through the login prompt without paying much attention to the login information. Further, Tap-2FA is susceptible to user negligence especially when the attacker attempts to log in around the same time as the user [25]. The attacker can drop the login notification corresponding to the users login session so that the user sees only one notification from the attacker's login session and is very likely to accept it thinking it is his own login session's notification.

Both these Watch-2FA schemes are also vulnerable to standard phishing attacks just like plain password-based authentication, where the attacker masquerades as a reputable entity, e.g., via a fake website, to steal the victim user's login credentials and later attempts to access the victim's account leveraging the stolen credentials.

Our Approach: We believe that the aforementioned Watch-2FA schemes do not really utilize the full potential and capabilities of the smartwatches, and are hence prone to these security and usability problems. In this paper, leveraging the unique characteristics of the watches (e.g., up/downstream online communication with the web service and onboard motion sensing) and addressing the above security-usability tension in current Watch-2FA systems, we introduce a fundamentally different design of minimal-effort Watch-2FA, called *SG-2FA*. The proposed system is based on the user's unique wrist motions, a simple sideways character typing gesture, called the "seamless gestures"[1], captured from a wrist-worn device. SG-2FA uses the knowledge of the password as the first factor of authentication (like other schemes), whereas wrist motions captured from the app installed in the user's wrist-worn wearable device is used as the second factor.

To make the seamless gestures distinguishable from other hand gestures exhibited in real-life, SG-2FA design carefully incorporates a *very short* sequence of special characters, which when typed associates uniquely the login attempt with the device's owner typing the sequence. This special sequence *does not need to be memorized (and is thus memorywise effortless [4]), or kept secret* by the user, but could rather be shown to the user on the browser, eliminating the need to interact with the watch and diverting attention away from the browser. It can also be fixed (constant) across multiple sites deploying SG-2FA (e.g., "@$@$"), as visualized in Figure 1c.

Via simple inspection, it is clear that the usability of SG-2FA over PIN-2FA would be better since only a short fixed sequence has to be typed as part of the authentication process (interacting with the wearable device and/or copying random PINs to the web browser are not needed). At the same time, the security of SG-2FA compared to Tap-2FA is significantly better. Unlike Tap-2FA, there is no reliance on user diligence, and user negligence does not translate into successful attacks. Moreover, SG-2FA is secure against traditional phishing attacks, in contrast to both PIN-2FA and Tap-2FA [20]. In fact, in SG-2FA, the attacker can not login on behalf of the user *unless* the user's wrist is undergoing *the exact same seamless gestures at the exact same time*. ***SG-2FA is designed to offer: (1) better usability compared to PIN-2FA (not better security) and (2) better security compared to Tap-2FA (not better usability).***

Although SG-2FA employs user's wrist gestures for authentication, it is ***not a biometric scheme*** because SG-2FA does not rely on the user-intrinsic wrist movements. This is a significant advantage in terms of privacy as no sensitive information about the individual user needs to be stored on the server. Although the seamless gestures is not unique to an individual user, it is different from other user activities such as walking, standing, playing and typing that enables SG-2FA to use the seamless gestures for second factor authentication. Due to the non-unique nature of the seamless gestures, unlike behavioral biometrics, SG-2FA does not need to build/train a separate prediction model for each individual user – a generic model is sufficient.

Our Contributions: Our work brings forth the following contributions to the field of web authentication:

1) ***New Wearable 2FA Notion based on Seamless Gestures:*** We introduce the idea of strong, privacy-preserving and low-effort wearable (wrist-worn) second factor authentication based on the notion of seamless gestures, giving rise to a concrete instantiation, the SG-2FA system.
2) ***Design and Implementation of SG-2FA:*** We design and implement SG-2FA for the Wear OS and the Chrome browser. Our design is based on machine learning tech-

[1]Short, EAsy and Memorywise effortLESS gestures

niques, particularly the Random Forest classifier, which captures seamless gestures based on a total of 144 features extracted from the recordings of watch's motion sensors, accelerometer and gyroscope (sampled at a limiting rate of 200 Hz), embedded in the watch. The seamless gestures is a simple back-and-forth lateral wrist gesture generated when typing a special sequence (e.g., @$@$).

3) ***Evaluation in Benign and Adversarial Scenarios:*** We evaluate SG-2FA for authentication errors in both benign and adversarial settings based on the wrist motion data collected from 30 volunteering participants while they input special sequence during the login process. Our results show that SG-2FA can effectively identify the legitimate users and block the adversaries with high accuracy (F-Measure of 99.29% with a special sequence of four character length) based on appropriate parameterizations. Moreover, the study participants perceived our SG-2FA system to be acceptable for use based on the system usability score (i.e., 73.46). Further, we find that the seamless gestures can be entered and captured quickly, within just 2 seconds.

Summary of the Main Result: Overall, our work shows that SG-2FA allows the user to login with low cognitive effort and offers a high level of security against a wide variety of attacks. It essentially forces the attacker to login precisely at the same time as the user logs in and even then the attacker can not succeed with a high probability. In occasional scenarios, where the legitimate user is denied access due to detection errors, SG-2FA can simply *fall-back* to PIN-2FA, thereby offering seamless integration with currently-deployed technology, and still keeping the process low-effort on most occasions and nearly as secure as PIN-2FA. In sum, all of these features place ***SG-2FA at the security level similar to that of PIN-2FA and the usability level similar to that of Tap-2FA.***

II. Our Approach

A. Overview

We consider a laptop/desktop browser-based authentication system to a remote web-server ($\mathcal{WS}$) and introduce a minimal-effort Watch-2FA scheme, *SG-2FA*. In SG-2FA, the knowledge of password constitutes the first login factor while the user's wrist-worn wearable device, particularly smartwatch ($\mathcal{W}$), constitutes the second factor. SG-2FA verifies the user's possession of the second factor based on the user's wrist gestures generated while providing a short special sequence. Although 2FA (including SG-2FA) is mostly used for web-authentication, SG-2FA can be used to authenticate the user to a laptop or desktop PC.

We consider a smartwatch personal to the user that features Internet connectivity (either directly or via a companion device like a smartphone as the proxy) and onboard motion sensing. SG-2FA requires the user to wear such a watch during the login process like any other Watch-2FA. It also requires the user to install an application/token in $\mathcal{W}$, and link to his account on $\mathcal{WS}$. This one-time operation can be carried out using existing techniques, for instance, the one used in Google

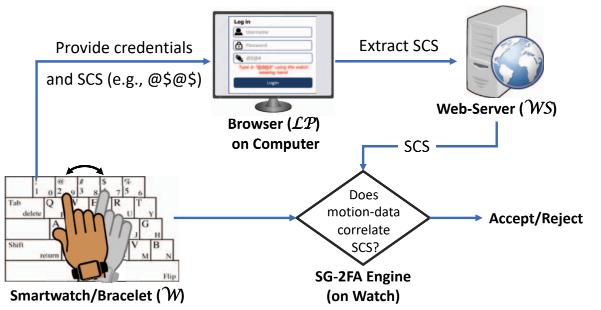


Fig. 2: A high-level overview of SG-2FA authentication. At login, the smartwatch/bracelet ($\mathcal{W}$) captures the unique wrist movements "seamless gestures" while the user inputs a special character sequence (scs) shown on the browser ($\mathcal{LP}$) during the authentication process. $\mathcal{W}$ (SG-2FA Engine to be specific) correlates the wrist movements with scs and makes the decision.

2-Step Verification [17]. The token installed in $\mathcal{W}$ records the wrist motion readings when the user provides a short special sequence. $\mathcal{W}$ then decides whether to accept or reject the current login attempt based on captured wrist motion readings. A high-level flow of SG-2FA is shown in Figure 2.

In SG-2FA, during the login process, the login web-page ($\mathcal{LP}$) of $\mathcal{WS}$ generates a short and simple *Special Character Sequence* (scs, details are provided later in Section IV) and displays it to the user. SG-2FA requires the user to provide his credentials, i.e., his username and password, and scs (shown on $\mathcal{LP}$) at the time of login. The typing of scs (e.g., "@$@$") creates a unique wrist-signature, specifically a simple sideways character typing gesture, termed as "seamless gestures", which is used to authenticate the user. SG-2FA extracts the motion-segment corresponding to scs from $\mathcal{W}$. If SG-2FA succeeds to verify the user's credentials as well as the wrist motion-segment associated with scs, i.e., seamless gestures, SG-2FA accepts, otherwise, rejects the login attempt. At its core, SG-2FA authenticates the user by verifying that the correct credentials have been supplied and scs has been typed while wearing the watch.

The current web-authentication systems often provide the "Remember Me" feature to improve its usability, where the password is filled automatically. Password managers are also often used alongside the web-authentication to enhance the security of the use of passwords while alleviating the cognitive burden of remembering those passwords by storing them in a secure encrypted vault. Similar to "Remember Me" feature, password managers also automatically fill the password for the user. In such a scenario, the user simply needs to perform second authentication process by simply typing in a short scs.

B. Comparison with Deployed Schemes

SG-2FA provides unique security and usability advantages compared to prior well-known and deployed Watch-2FA schemes, i.e., PIN-2FA and Tap-2FA, which we describe below. Figure 3 presents a brief summary of the security-usability comparison of SG-2FA with PIN-2FA and Tap-2FA.

Appendix E Table VII expands on this comparison using the classical framework of Bonneau et al. [4]

Compared to PIN-2FA, SG-2FA neither requires user-watch interaction nor diverts user's attention from the PC browser, mere typing of a short sequence shown on the browser is sufficient to login. Unlike PIN-2FA, SG-2FA does not require the user to open the authentication app on the watch and interact with it. Therefore, the small-form factor of the watch (that may negatively impact the usability of PIN-2FA) does not have any effect on the usability of SG-2FA. Since SG-2FA does not involve the verification code, unlike PIN-2FA, SG-2FA is memorywise effortless (except of the password, the user does not need to memorize anything). Further, SG-2FA incurs a relatively small amount of time (~5 seconds) compared to PIN-2FA ($>$10 seconds) during the authentication process [22], [30]. Thus, we believe that it is rather obvious that the usability of SG-2FA will be better compared to PIN-2FA. Moreover, SG-2FA is secure against standard phishing attacks to which traditional 2FA schemes are vulnerable as shown in [20]. In such a phishing attack, an attacker gains the OTP code generated by the user's watch by faking an ongoing malicious activity on the user's account and asking the user for the recently generated verification code. When launching a traditional phishing attack against SG-2FA, an attacker lures the user into visiting a phishing website and relays the stolen credentials to the attacker that he supplies to the legitimate website in real-time. During such phishing attacks, as the attacker's typing of scs will not fully overlap with that of the victim user, SG-2FA can effectively detect and prevent such attacks.

Tap-2FA requires the user to simply tap on a login attempt notification shown on the watch for user's approval. In Tap-2FA, the user is supposed to read and verify the login information shown on the login notification. Therefore, Tap-2FA is not completely memorywise effortless. Further, the approach still requires the users to interact with the watch, which may divert their attention away from the primary point of work, the PC browser. Moreover, this approach is susceptible to potential user errors or negligence [30], as users may simply tap/approve notifications (including the fraudulent ones) without paying attention (*click-through behavior and habituation* have already been widely highlighted in user-centered security literature, e.g., [13], [44]). In particular, if the attacker attempts to log in around the same time as the user logs into his account, while dropping the user's push notification, user may approve the attacker session's notification thinking that he is approving his own session's notification [25]. Tap-2FA is also vulnerable to traditional phishing attacks, where the attacker lures the user to log into a phishing website, and around the same time attempts to login. The victim is highly likely to accept the attacker's generated login notification thinking that he is approving his own session's notification. Unlike Tap-2FA, the security of SG-2FA does not rely upon the actions or decisions of the end users. In fact, an attack against SG-2FA cannot succeed until and unless the time of typing the scs and corresponding inter-key wrist movements, i.e., $scs\text{-}motion$, of the attacker match

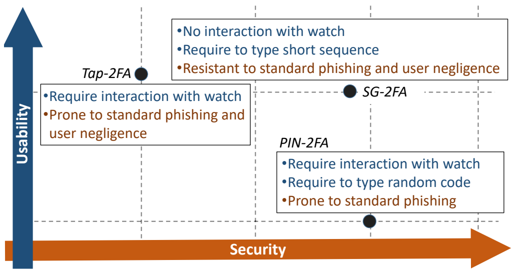

Usability
Security
Tap-2FA
•Require interaction with watch
•Prone to standard phishing and user negligence
•No interaction with watch
•Require to type short sequence
•Resistant to standard phishing and user negligence
SG-2FA
PIN-2FA
•Require interaction with watch
•Require to type random code
•Prone to standard phishing

Fig. 3: Security-Usability analysis of three wearable-based 2FA schemes: PIN-2FA, Tap-2FA, and SG-2FA.

with that of the victim user. However, it is hard to achieve such overlapping in practice (even with a phishing attack, as explained above) since the duration of login activity is short (just a few seconds).

C. *User Experience of Special Sequence*

The full formal usability study of SG-2FA is beyond the scope of this paper and could be pursued in future work. However, the user perception of using the special sequence in SG-2FA may be considered to be in line with the one in the studies involving the special sequence. For instance, similar to SG-2FA, the addition of special sequence, particularly prefix, to the password has been used in other security systems such as PwdHash [31] and SPHINX [39]. However, their purposes of using a special sequence with the password are different from our SG-2FA scheme. Unlike SG-2FA that uses the special sequence to generate a unique wrist gesture, PwdHash and SPHINX use the special sequence to activate their security schemes. The studies of [31], [39] showed that the users are generally comfortable with adding a short special sequence to their password. This result will apply to SG-2FA as well. Further, PwdHash has been shown to have the usability issue of the users forgetting to type the sequence during their login attempts that expose them to a phishing attack [7]. However, in case of SG-2FA, it would be very rare for the user to forget typing the sequence since the sequence is displayed on the browser itself that always reminds them that they have to type the sequence. Even if the user forgets to enter the sequence, it does not expose the user to any sort of attacks, it just results in an occasional false rejection (in which case the user can attempt the login again with the special sequence added).

III. Threat Model and Attack Settings

Our threat model assumes that all the communication between the devices involved in SG-2FA, i.e., $\mathcal{LP}$–$\mathcal{WS}$ and $\mathcal{WS}$–$\mathcal{W}$, are protected by cryptographic mechanisms (e.g., SSL/TLS) as is done in typical settings. It considers an adversary who has gained the victim user's login credential through phishing attacks, password database leakage, or other mechanisms. With the knowledge of user's login credentials, the adversary attempts to access $\mathcal{WS}$ from a remote machine on behalf of the victim user. The attack is successful if he can prove the possession of $\mathcal{W}$, which may be possible if the victim user happens to be typing the scs, or forced to create the wrist gesture close to typing scs, at the time the attacker types the scs, however, is a very unlikely scenario.

Like other 2FA schemes, we assume that the adversary can neither gain the physical access of nor compromise the second factor device, i.e., $\mathcal{W}$ in our case. If the adversary obtains the possession of (or has compromised) $\mathcal{W}$ designated for SG-2FA then the security of SG-2FA reduces to the security of password-only authentication system similar to other 2FA schemes. We also assume that the adversary cannot compromise the victim's PC browser. If the adversary compromises the victim's browser, then he can launch a man-in-the-machine attack and hijack the victim's session with $\mathcal{WS}$ thereby defeating any 2FA mechanisms.

We further consider a *random (untargeted)* remote attacker who attempts to login at random on behalf of the victim user. We also consider a targeted attacker who attempts to login by forcing the victim user into creating the wrist gesture close to scs. In the context of web-authentication, the remote attackers are more prominent and possess a higher level of threat compared to that of the targeted attackers. Therefore, we consider a random remote attack setting as a *high* likelihood threat to SG-2FA while a targeted attack setting as low likelihood threat to SG-2FA. Based on the attack setting and various activities that the victim user could potentially be performing during the attack, we categorize the potential threats to SG-2FA into three classes as follows.

❶ Threat 1 – Regular Wrist Movements: At an arbitrary time when a random passive attacker attempts to login, the user may be performing everyday regular activities such as walking, standing, resting on a chair, keeping the wrist static inside the pocket or on the desk, typing or playing a game on a phone, etc. These are the activities that the users are most likely to perform during their day-to-day lives. We consider that the wrist movements during these activities as *regular wrist movements*. Since these activities are performed most frequently by the user and random remote attackers are more prominent, we consider that the regular wrist movement activities present a *high* likelihood threat to SG-2FA.

❷ Threat 2 – Text Typing: The user may be using his computer or laptop (terminal) for official or personal use when a passive remote attacker attempts to login. An active targeted attacker may also force (through a phishing attack, targeted ads) the user to use/type on the terminal at the time he attempts to login. Since the typing activities are relatively less often performed by the users compared to their regular activities and the attack could be either a passive remote attack or an active targeted attack, we consider the wrist movement associated with text typing as a *medium* likelihood threat to SG-2FA. Out of various activities involved while accessing the terminals such as playing games, browsing using only the mouse, the activity that may closely match with scs typing is the activity of using the keyboard, i.e., the text typing. So, instead of considering a weak threat model, where victim user is executing a regular terminal accessing activity when an adversary attempts to login, we consider relatively strong threat model, where the user performs continuous typing on his terminal.

❸ Threat 3 – Password Typing: We consider a strong attack setting where a login attempt from an attacker (remote or targeted) overlaps with the user's login attempt. We assume that an active targeted attacker can fool the user into typing his credentials through a fake website (i.e., a traditional phishing attack). Since the attacker has a full control over the phishing site, he can get the victim to type his password multiple times, and coax the victim to type the password at the attacker's choosen time by controlling the loading time of the site. We also consider an insider attacker, potentially a disgruntled friend, or colleague, who knows the user's login habit and attempts to log in at the time when the user tries to log in. However, the login activity is of short duration (merely a few seconds) and is not performed very frequently. It is very unlikely that an adversary's login attempt and user's login attempt would overlap in a *real time*. ***Even if these login attempts are overlapped, the attacker needs to match his scs typing gesture, specifically corresponding inter-key wrist movements, i.e., scs-motion, with that of the victim user.*** In other words, simply matching the start of the scs typing with that of the victim user is not sufficient to break the SG-2FA scheme, it requires the exact matching of subsequent time series corresponding to the key press events while typing the scs, which is hard to achieve in practice. Therefore, we consider the attack setting with such overlapping poses a *low* likelihood threat to SG-2FA.

IV. System Architecture and Details

In this section, we introduce the *Special Character Sequence* (scs) used in SG-2FA to verify the second-factor device and present the concrete steps followed in our design of SG-2FA.

A. Special Character Sequence (SCS)

As mentioned earlier, in our SG-2FA scheme, the user needs to type in the scs (displayed on $\mathcal{LP}$) during the authentication process to generate the simple gesture password. Initially, we tested if password entry alone creates a motion signature that can be differentiated from user's all other activities. However, with our design, we found a high matching score in wrist-motion between the password entry and the regular text typing, and cannot be used for authentication purpose. Therefore, we proceeded with the addition of a short scs on top of the password in our SG-2FA to incorporate a motion signature on the password entry. There are certain rules that $\mathcal{LP}$ follows to generate the scs and simple practices to type the scs. They are described below.

SCS Generation Rules: The scs is formed using only the special characters located on the left side of the standard QWERTY keyboard. Specifically, seven special characters – {˜ , !, @, #, \$, %, ˆ} – found on the left side of the keyboard are used to form the scs. Two characters from this list of special characters are chosen and used to generate the scs by

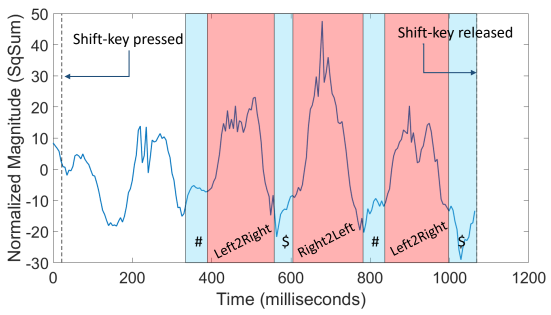


Fig. 4: Acceleration of user's wrist when he types scs– "#$#$". Left2Right – left-to-right; Right2Left – right-to-left wrist movement.

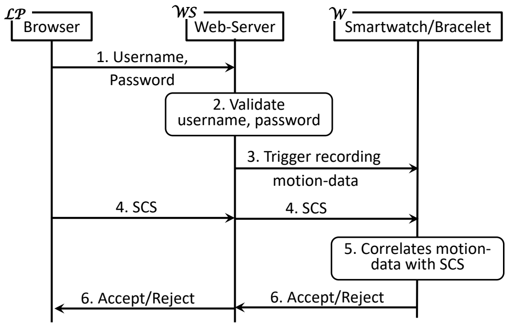
placing them alternate to each other. The alternate placement of characters in scs enforces to create a unique wrist motion that corresponds to either *left-to-right* or *right-to-left* wrist movement. If α and β are the two special characters, they can form two scs, one starting with α, i.e., $\alpha\beta\alpha\beta$, and another starting with β, i.e., $\beta\alpha\beta\alpha$. We consider two parameters while generating an scs for SG-2FA. First, the *length* that is the number of characters in the scs. Second, the *distance* which is the number of keys between two keys that forms the scs. The scs can have three different character lengths – *Len3, Len4, and Len5* for three, four and five character lengths, respectively. Regarding the inter-key distance between two special characters (α, β) forming the scs, we consider four different distances – *Dist0* if α and β are side-keys, *Dist1*, if there exists one key between α and β on the keyboard, *Dist2*, if two keys exists between α and β, and *Dist3*, if there exist three keys between α and β.

Simple Typing Rule: As all the special characters are located at the left side of the keyboard, only left hand wearing $\mathcal{W}$ device should be used to type the scs so that it can capture the corresponding wrist motion. Since all the special characters require the "Shift" key to be pressed, while typing the scs during the password entry, the user can use any of the two Shift keys (Right-Shift and Left-Shift) available on the keyboard. We note that no special training is required to type the scs, mere typing with one hand is sufficient.

Although, in our study, we design and evaluate SG-2FA geared for the users who wear the watches/bracelets on their left hands (a commonly occurring setting), it can be setup alternatively for users who wear their wrist devices on their right hands. Instead of using the special characters at the left side of the keyboard, special characters at the right side of the keyboard can be used to form the scs. The right wrist motion of the users can now capture the gesture associated with scs. To identify whether to leverage the left-handed scs or the right-handed scs, during the registration phase, SG-2FA requests the user to specify whether he would use right- or left-hand to provide scs.

SG-2FA Seamless Gesture: Figure 4 shows the acceleration of user's wrist when he types the scs– "#$#$" (corresponding gyroscope signal is shown in Appendix A Figure 9). The thinner shaded region in the figure shows the key-pressed events, and inter-key wrist movements, i.e., the wrist movement from *left-to-right* or *right-to-left*, are represented by the thicker shaded regions. We denote both of these inter-key wrist gestures by scs-$motion$. As shown in the figure, when the user moves his wrist to type two consecutive keys of scs, it raises the amplitude of motion signal to a certain level that we use for the authentication purpose in SG-2FA.

We note that the SG-2FA gesture varies based on the *layout* of the keyboard, specifically the location of the special characters used to create scs, but does not change based on the model/manufacturer of the keyboard, specifically the physical hardness of the keyboard. Since most of the standard keyboards follow a similar layout, changing the model of the keyboard with different hardness (e.g., the laptop keyboard) does not have much effect on SG-2FA.



$\mathcal{LP}$ Browser
$\mathcal{WS}$ Web-Server
$\mathcal{W}$ Smartwatch/Bracelet
1. Username, Password
2. Validate username, password
3. Trigger recording motion-data
4. SCS
4. SCS
5. Correlates motion-data with SCS
6. Accept/Reject
6. Accept/Reject

Fig. 5: SG-2FA authentication overview. At login, the user provides the special character sequence (scs) shown on the browser, and the smartwatch/bracelet ($\mathcal{W}$) captures the user's wrist-movement. $\mathcal{W}$ then makes decision by correlating the wrist-movement with the scs and relays the decision to the user via the web-server ($\mathcal{WS}$). The communication between $\mathcal{WS}$ and $\mathcal{W}$ happens via the phone as a proxy in our current implementation.

B. Concrete Steps of SG-2FA

The concrete steps followed in SG-2FA are outlined below. Figure 5 shows a visual outline of the steps.

Step 1 – 3: The user provides his credentials, i.e., his username and password, on $\mathcal{LP}$ and submits them to $\mathcal{WS}$. $\mathcal{WS}$ then verifies the validity of supplied credentials. If $\mathcal{WS}$ cannot verify the legitimacy of user's credential, it promptly rejects the login attempt. Once the user's credentials have been verified, $\mathcal{WS}$ triggers a token (a wear-app) installed on the watch to record motion sensor data. Then, $\mathcal{LP}$ generates and displays a short, simple and random scs to the user.

Step 4: The user provides scs on $\mathcal{LP}$, which records all the input events (with timestamps) at the scs field – *scs time information*. $\mathcal{LP}$ then sends the scs time information to $\mathcal{W}$ via $\mathcal{WS}$. In our current implementation of SG-2FA, we use a smartphone ($\mathcal{P}$) as a proxy between $\mathcal{WS}$ and $\mathcal{W}$. Therefore, $\mathcal{WS}$ first contacts $\mathcal{P}$, which in turn sends the time information to $\mathcal{W}$. Google Cloud Messaging (GCM) APIs [15], its newer version Firebase Cloud Messaging (FCM) APIs [16], or Apple Push Notification (APN) APIs [2] can be utilized to communicate with $\mathcal{P}$. The phone $\mathcal{P}$ has no role except of being a proxy in this implementation. In a real-world implementation with a watch having direct Internet connectivity, there will be no need for the phone.

Step 5: When $\mathcal{W}$ receives scs time information from $\mathcal{WS}$, it extracts the motion measurements ($scs\_segment_m$) corresponding to scs entry. $\mathcal{W}$ is embedded with a well-trained classification model, particularly scs-*Verifier*, that classifies the supplied $scs\_segment_m$ to scs entry or *non-scs* entry. If scs-*Verifier* succeeds to validate the $scs\_segment_m$, it accepts the login attempt, otherwise rejects it. The $\mathcal{W}$ device is generally resource constrained in nature and usually transfer all the computations to its companion device, i.e., $\mathcal{P}$. For such a $\mathcal{W}$ device, it can stream motion data to $\mathcal{P}$ and all decision making process (i.e., the scs-Verifier) can be implemented on $\mathcal{P}$. Since SG-2FA relies on the seamless gestures (not on the secret PIN or password), even if the exact scs sequence that the user is to type is leaked, compromised or otherwise known to the attacker, it does not lower the security of SG-2FA.

Step 6: $\mathcal{W}$ then relays the decision – "Accept or Reject" – to $\mathcal{WS}$, which in turn forwards the decision to the user (i.e., displays it on $\mathcal{LP}$).

During the login process, $\mathcal{LP}$ and $\mathcal{W}$ run the time-synchronization protocol with $\mathcal{WS}$ to adjust their local clock differences with $\mathcal{WS}$.

Fall-back Scenarios: There may exist some scenarios where SG-2FA may not be able to capture the user's wrist-motion when he logs in, and make SG-2FA unable to verify the legitimacy of the user. For instance, $\mathcal{W}$ may not record and process the motion readings required for verifying the possession of $\mathcal{W}$ as a second-factor. In such scenarios, the user can always fall back to PIN-2FA implemented using $\mathcal{W}$, i.e., copy the PIN from $\mathcal{W}$ to $\mathcal{LP}$ to prove the possession of $\mathcal{W}$ as a second-factor. We note that such fall-back is occasional and will not lower the security of SG-2FA.

Extensions to Phone based Web-Login or IoT Device Authentication: Unlike traditional phone-based PIN-2FA, our SG-2FA scheme can be extended easily to support the 2FA login from the phone. In such extended SG-2FA, the user needs to type the scs (displayed on the phone browser). The wrist gesture while typing scs using a soft keyboard on the phone could be different from the gesture while typing using a physical keyboard potentially because they differ significantly in their size. Further, unlike standard physical keyboards that are positioned fixed on the surface, phones are generally held by the users on their hands in various position and orientation. Therefore, to extend SG-2FA for the phone, a separate scs-*Verifier* needs to be built using the scs-*motion*s derived from typing scs on the phone itself. Moreover, in contrast to standard physical keyboards that have more or less similar dimension, the size of the soft keyboard on the phone varies with the size of the phone's screen that may impact on the wrist gesture used in SG-2FA. Besides our current seamless gestures, different other types of gestures, such as typing leftmost/topmost key and rightmost/bottommost key back-and-forth, swipping back-and-forth from left-to-right (or top-to-bottom) and vice-versa, can also be employed for SG-2FA. Such seamless gestures captured near transparently by a watch can also be used for various applications other than 2FA, such as authenticating the user to a local terminal, IoT devices, or even unlocking the phone device. Future research is needed to realize such extensions of SG-2FA.

Real-Time Phishing Attack: The attacker may utilize an advanced and real-time phishing technique, e.g., Evilginx [46], a man-in-the-middle attack framework, that captures credentials and session cookies of a web service on-the-fly to bypass the security of 2FA. However, stealing credentials and cookies are not sufficient in the case of SG-2FA. The attacker should be able to spoof the exact timestamps corresponding to scs typing in a real time, which is hard to achieve. The attacker can spoof such timestamps only by manually modifying the respective JavaScript code block, which would create significant delay that can be easily detected using a set threshold for the response. Further, to defeat such a real-time phishing attack, phishing prevention schemes, such as 2FA-PP [45] and Hacksaw [42] can be deployed with SG-2FA. 2FA-PP leverages Bluetooth API that enables the direct communication between the browser and the second factor device and network latency measurements to check if the user is indeed connected to the legitimate server. Although 2FA-PP is originally proposed for a smartphone, it can be extended to support a smartwatch as used in SG-2FA. Hacksaw, on the other hand, already uses watch to re-authenticate the user after he has logged into the system. It operates by correlating the sequence of events (e.g., key presses/releases) with the sequence of events predicted based on the wrist-motions captured by the watch. Hacksaw can be deployed only for a short duration at the beginning of the session.

V. Design and Implementation

As for a prototype design and implementation, and later for the testing of SG-2FA, we used Sony Smartwatch 3 SWR50 [43] as a second-factor device, and a desktop PC with an external US keyboard as a terminal for the user to simulate the login scenario with SG-2FA. The Sony watch contains accelerometer and gyroscope sensors. Although only a standard US keyboard was considered in our study, SG-2FA can be easily extended to support other keyboard layouts (e.g., UK, French, German) because scs can be formed using any two special characters located on the left region from any location of the keyboard for left-handed wearers (and right region of the keyboard for right-handed wearers). Our implementation of SG-2FA consists of following components. Figure 6 depicts our prototype implementation of SG-2FA.

❶ **Website and WebServer:** We implemented a website consisting of a simple login form using HTML, Javascript, CSS, and PHP. The website was hosted in a XAMPP web-server. The web-server uses Google Cloud Messaging (GCM) to communicate with the designated smartwatch. Our website records all the input events on the keyboard. Specifically, it records the ASCII value of the pressed key, its timestamps

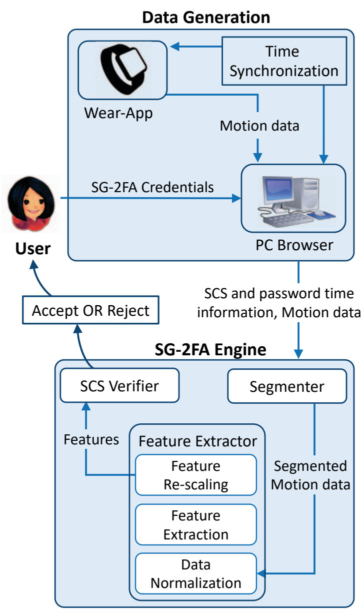


Fig. 6: SG-2FA Prototype Architecture.

when the system dispatches KEY_PRESSED and KEY_RELEASED events. It also records the timestamps when the user starts and ends typing his password. All these recordings are uploaded to the web-server for the purpose of offline analysis in our current implementation.

❷ **Watch Application:** We built a *Wear-App* for an Android smartwatch. The wear-app installed on the watch records and streams the motion signals, i.e., accelerometer and gyroscope readings, to the web-server. These two sensors capture the device/wrist movement, such as the movement from left-to-right (or right-to-left). We note that SG-2FA uses the standard sampling rate (i.e, 200Hz) of the current generation of smartwatches. Further, SG-2FA records and processes motion signals for only a short period of time (<6 seconds). Therefore, power consumption will not be an issue for SG-2FA.

❸ **Time Synchronizer:** As the terminal loading the website and the watch running the wear-app may have different local clocks, SG-2FA requires recordings from these two to be synchronized. For this reason, they run a simple time synchronization protocol with the web-server while they are collecting their respective data. This allows the terminal and the watch to compute the time difference between the local clocks and the one on the web-server. During the analysis, the recordings from these devices are adjusted accounting the clock difference with the web-server.

❹ **SG-2FA Engine:** This unit is the core component of SG-2FA, which is responsible for verifying the seamless gestures. It consists of following three components.

• ***Segmenter:*** Segmenter receives the password time information, the series of input key events, and the motion sensor readings. The password time information takes the form of $(t_{pswd\_strt}, t_{pswd\_end})$, where t_{pswd_strt} is the timestamp when the first character of sg_pwd is pressed and t_{pswd_end} is the timestamp when the last character is released after being pressed. Input key events data take the form of:

$$(key\_code_i, t_{pres\_i}, t_{relz\_i}), (key\_code_j, t_{pres\_j}, t_{relz\_j}), ..,$$

where key_code_i is the ASCII value of the key pressed at time t_{pres_i} and released at time t_{relez_i}. The accelerometer data is of the form:

$$(t_i, x_i, y_i, z_i, m_i), (t_j, x_j, y_j, z_j, m_j), ...,$$

where $(t_i, x_i, y_i, z_i, m_i)$ represents one acceleration data sample recorded at time t_i and x_i, y_i, z_i are instantaneous acceleration along x, y and z axis, respectively. m_i shows the magnitude $(SqSum)$ of the data sample at time t_i, which is computed by $\sqrt{{x_i}^2 + {y_i}^2 + {z_i}^2}$. The gyroscope data is of the similar form.

Since the motion sensor measurements vary even for a small tilt and shift, the readings from each axis of accelerometer and gyroscope, are first mean-normalized (i.e., $raw\ values - mean$) so that the readings fluctuate around zero. Segmenter then extracts $pswd\_segment_i$, the input-event segment from the terminal that lies within the time frame of $[t_{pswd\_strt},\ t_{pswd\_end}]$. Within $pswd\_segment_i$, Segmenter looks for the presence of scs based on the possible ASCII key code combination for scs. If Segmenter does not find any scs within the $pswd\_segment_i$, it rejects the current password entry. Based on the time when scs starts and ends, Segmenter extracts $scs\_segment_i$ (i.e., the *input* event segment corresponding to scs) and $scs\_segment_m$ (i.e., the *motion* segment that corresponds to scs entry).

Since scs is composed of multiple special keys, it consists of several scs-$motion$ (*right-to-left* or *left-to-right*) segments depending on the length of scs. Specifically, an scs consists of $(n$-$1)$ scs-$motion$, where 'n' is the length of scs. Based on the key released timestamp of one key and key pressed timestamp of the subsequent key in $scs\_segment_i$, Segmenter extracts $(n$-$1)$ scs-$motions$ from $scs\_segment_m$, and feed them to Feature Extractor.

• ***Feature Extractor:*** It receives the motion measurements from Segmenter in the form of blocks, i.e., the series of scs-$motion$. Feature Extractor extracts various features from each of the scs-$motion$ (details provided in Appendix B Table II). Specifically, it computes 18 features – *minimum, maximum, mean, median, standard deviation, variance, median absolute error, inter-quartile range, power, energy, spectral entropy, autocorrelation, kurtosis, skewness, median frequency, peak counts, peak-to-peak amplitude, and peak-magnitude-to-rms ratio* – over each series of accelerometer and gyroscope measurements in an scs-$motion$. We chose these features because others have used them successfully for activity recognition [29] and terminal interaction prediction [26]. In total, Feature Extractor computes 144 feature vectors from each scs-$motion$. The feature values were rescaled in the range of [0, 1] with Min-Max scaling [28]. The scaled feature vectors from each scs-$motion$ are then supplied to scs-verifier.

• ***SCS-Verifier:*** It is a critical component of SG-2FA Engine that is responsible for deciding whether to accept or reject the current login attempt by validating the scs based on the scs-$motion$ gesture. scs-*Verifier* consists of a well-trained classifier for predicting scs-$motion$ based on the motion measurements. Based on the feature vectors received from Feature Extractor, the classifier of scs-*Verifier* predicts if scs-$motion$ segment is associated with an scs-$motion$ or not. Based on the number of successful association/verification of scs-$motions$ corresponding to an scs, scs-*Verifier* decides whether to accept or reject the login attempt.

As a classifier for scs-*Verifier*, we tested several state-of-the-art machine learning algorithms, e.g., Multilayer Perceptron, Random Forest, and Support Vector Machine (performance of the different classifiers is presented in Appendix B Table III). As Random Forest outperformed all the other classifiers, we chose Random Forest as a classifier for scs-*Verifier*. Prior to training the classifier, we applied one of the supervised instance reduction techniques, namely spread subsample filtering [47], on the training samples to produce a well-balanced training dataset and remove the biases that an unbalanced dataset may create. To train the classifier, we feed it with the feature vectors extracted from the user's wrist motion segment and provide the actual labels. The label '1' represents scs-$motion$ while the label '0' represents the activities other than scs-$motion$. We note that the features associated with the user's wrist motion segment are not unique (rather general) to an individual and do not correspond to biometrics.

VI. User Study Design and Data Collection

For our data collection, we recruited 30 voluntary users at our University. In our experiment, majority of the participants were 25-35 years old (26) and right-handed (28); 22 male and 8 female. Most of the participants who own a wrist device are used to wearing the device on their *left-hand*. The demographics of the participants and their wrist device usage are summarized in Appendix C Table IV. Similar number of participants and demographics are well-established in lab-based studies in behavioral biometrics research [9], [37], [49], [48], which serves to demonstrate the viability of the schemes.

Prior to the experiment, participants were told that the purpose of the experiment was to study the feasibility of using wrist gestures while providing the password to authenticate the user. They were informed that the wrist-gestures, particularly accelerometer and gyroscope data, and keyboard input events will be recorded. The experiment and data collection followed the IRB procedures at our institution.

At the beginning of the experiment, the users were provided with a pool of passwords that were generated using SAFEPASSWD [35], an online password generator. Our pool of passwords consists of 100 unique, strong (marked by SAFEPASSWD) but easy-to-remember passwords with the character length ranging between 10 and 14. We asked the users to choose one of the passwords from the pool that they used during the entire experiment.

Next, the participants were given the instruction about the scs generation and scs typing rules as mentioned in Section IV. Each participant went through a practice session, where they were asked to pick a random scs, prefix it to the chosen password to generate sg_pwd, and practice typing sg_pwd for multiple times. The purpose of this practice session was to make them familiar with the use of sg_pwd, and with the SG-2FA in general. In the real-world scenario, the user can use (prefix or postfix) the scs shown on $\mathcal{LP}$ to generate sg_pwd. Further, no special training is required to type the scs, mere typing with one finger is sufficient.

The participants were then asked to create several scs considering the combination of scs's character lengths and key distance. In total, they were required to create 12 ($3 * 4$, three scs lengths, and four key distances) different types of scs. For each of the key distance and scs length setting, they were allowed to choose different two special characters. In our study, most of the times, the participants chose (@, #) or (@, !) as two special characters to form the scs for *Dist0*, while for *Dist1*, they chose (@, $) or (!, #). Similarly, for *Dist2*, they chose (!, $) or (@, %), while they chose (!, %) or (@, ^) for *Dist3*. Appendix D Table V shows the list of scs picked by our study participants. They were instructed to *prefix* each of the generated scs to the original password they have chosen earlier to generate 12 different sg_pwd for SG-2FA. Although in real-world implementation of SG-2FA, it generates the scs by itself and displays it on $\mathcal{LP}$, we asked users to create scs of their choice so that they become familiar and comfortable using scs of their choice. Further, SG-2FA can work with a user-chosen scs that complies with the scs properties for flexibility purposes (it does not necessitate a system-chose scs), hence we allowed the participants to follow this option, which also enabled us to capture scs patterns and timings (Appendix Table V and VI).

Using each sg_pwd, the participant registered to our prototype implementation of SG-2FA. The participants were then required to login with that particular sg_pwd successfully for 10 times. Since the majority of right-handed users usually wear watch/bracelet on their left-hands (a more common setting, applicable to our participants also), we tested our SG-2FA with the watch being worn in the left-hand settings (we discussed how SG-2FA can be extended to support right-handed wearers in Section IV-A). We asked the participants to wear the watch on the wrist of their left hand during the entire study.

We repeated the experiment with each participant three times. The time gap between two consecutive sessions ranged between (1-10) days. The order of the experiment for different scs lengths to each user followed 3x3 Latin square. Each session took approximately 15-20 minutes. In sum, we collected 36 data samples (360 correct password entries) per user. On average, participants took nearly six seconds to type a sg_pwd and less than two seconds to type an scs. The detailed results are presented in Appendix Table VI.

At the end of the study, we asked the participants to rate the usability of SG-2FA answering the SUS (System Usability Scale) questionnaire [5]. SUS is a standard and reliable tool that is frequently used to measure the usability of software systems. The SUS survey contains 10 standard usability questions, each with five possible answers (5-point

Likert scale, where 1 represents strong disagreement and 5 represents strong agreement). The mean SUS score for our SG-2FA was 73.46 ($\pm$8.90), which is representative of above average ($>$68) [5] and good ($>$70) [3], [33] usability.

We also collected data samples for each of the victim activities mentioned earlier in Section III from randomly selected 2-5 participants to evaluate our SG-2FA against the attack setting. We asked two participants to walk for three minutes in their regular walking style. Similarly, we asked another two participants to stand or rest/sit on a chair for three minutes and allow them to move their hands in a natural way. For two of the participants, we asked to play a game and type some text provided to them on a phone. We asked two participants to keep their hands still on the desk and inside the pocket. Five participants were provided with some wiki links and were asked to type the text on a notepad for five minutes while we asked all the 30 participants to pick and type three passwords (including scs) from the password-pool, three times each. While performing these activities, we asked the users to wear the watch to collect the motion sensor readings corresponding to these activities. Thus, we collected two data samples for each of the following regular activities – walking, sitting on a chair, playing a game on a phone, and typing text on the phone, five data samples for continuous text typing, and 30 data samples (each containing 9 password entries) for password typing.

VII. Performance and Security Analysis

A. Evaluation Preliminaries

When evaluating our SG-2FA, particularly scs-*Verifier*, we employ *Leave-One-Subject-Out (LOSO)* approach. In particular, for a given user, we build the classifier using samples from all other users. *The use of such an approach indicates that the SG-2FA system does not require any user-specific prediction model, thereby makes the model user-agnostic and results in a generic, user-independent classification model for scs-Verifier.* In the real world implementation, scs-*Verifie*r can be trained using the data samples from either by recruiting a set of certain users or the people from the developer ends. It does not require to collect data samples from each individual user of the system. While applying the LOSO model, we build a separate classifier for each of the key-distance under consideration to improve the performance of the scs-*Verifier*.

Performance Metrics: scs-*Verifier* of SG-2FA is bi-nominal in nature. During the classification, data instances from the users when performing the login task correspond to the positive class and the data instances when performing activities other than password-entry correspond to the negative class. Further, the user may be performing any tasks when an adversary attempts to login on behalf of the victim user. Therefore, False Rejection (FR) indicates the number of times a login attempt from the legitimate users is classified incorrectly as non-legitimate attempts, and False Acceptance (FA) indicates the number of times a login attempt from the attacker is incorrectly classified as legitimate login attempt. We use False Rejection Rate (FRR) to measure the performance of SG-2FA (scs-*Verifier* in particular) in benign settings, i.e. while authenticating a user. FRRs in our evaluation are independent, i.e., the repeated attempts for authentications are not correlated in terms of FRR. To measure the performance of scs-*Verifier* in adversarial settings (as described in our threat model in Section II), we use False Acceptance Rate (FAR). We also use precision, recall, and F-measure (F1-Score) to measure the overall performance of SG-2FA. Precision measures the security of the proposed system, i.e., the success rate to reject the impersonators. Recall measures the usability of the proposed system, i.e., the accuracy of accepting legitimate users. F1-score is the harmonic mean of precision and recall. To make our SG-2FA scheme more accurate, we would like to have low FRR, low FAR with high values for precision, recall, and F1-score.

Error-Threshold: It is the number of mis-predictions allowed while inferring the scs based on the predictions of scs-$motion$ segments. Given scs of length 'n', we consider three different error-thresholds for scs-*Verifier*.

- *All-Correct:* This threshold setting requires all scs-$motions$ of scs to be predicted correctly.
- *One-Error:* One mis-prediction out of $(n-1)$ predictions of scs-$motions$ is allowed in this setting.
- *Two-Error:* This threshold setting allows two mis-predictions out of $(n-1)$ scs-$motion$ predictions.

All-Correct and One-Error are applicable for all lengths of scs (*Len3*, *Len4*, and *Len5*) while Two-Error is applicable only for *Len5*. Higher the error-threshold, more flexible and usable will be the scs-*Verifier*, the SG-2FA as a whole. However, the high error-threshold may also make the system vulnerable and less secure. The analysis on the impact of error-threshold on the performance of SG-2FA in benign and adversarial settings is presented next.

B. Results

1) Benign Setting: To evaluate the performance of SG-2FA in a benign setting, we employ the LOSO cross-validation approach – for each user, we train the Random forest classifier using 87 data samples (870 password entries) from other users as positive instances. The training samples also contain negative instances, which are generated as follows. To create the negative instances for the training dataset, the $scs\_segment_i$, i.e., the input segment associated with scs, from each of the users is cross-mapped to the wrist-motion from at least one of the attack activities and features were extracted from the cross-mapped $scs\_segment_m$. To evaluate the classifier, we then use three data samples (30 password entries) from the current user. For each key-distance, we thus train 30 different classifiers and report the results aggregating classification of 90 data samples (900 password entries).

Figure 7 shows the FRRs for various key-distance and lengths of scs. The first graph block shows the FRRs when the error threshold is set to All-Correct while the second block and the third block show the FRRs when the error threshold

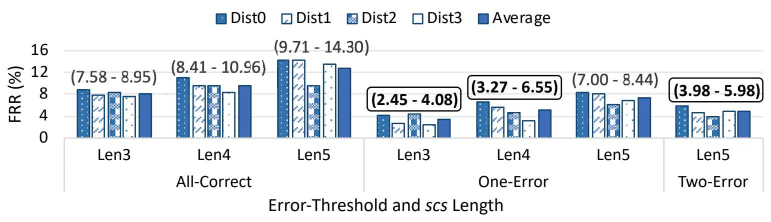


Fig. 7: Benign Setting. False Rejection Rate (FRR) of SG-2FA for various scs length and scs key-distance setting with three different error-threshold settings. "Average" bar on each scs length setting shows the average of FRR over all the key distances.

is set to One-Error and Two-Error, respectively. In the All-Correct setting, the average FRRs (last bar in each block of each *scs* length) ranged between (8-13)%. Relaxing the *scs-Verifier* of SG-2FA by setting the error-threshold to One-Error, the average FRRs got improved to the range of (3-8)%. When SG-2FA was set to Two-Error for *Len5*, the average FRR dropped to 4.90% from 7.41%. We did not find any significant difference in the performance (in terms of FRR) of SG-2FA based on the key-distance of the special character pair used to generate *scs*. All the key-distances considered in our evaluation equally performed well. All these FRRs are based on independent login attempts. The scenario where the legitimate user re-attempts to login upon failure from the initial login is not considered in our FRR computation.

Overall, SG-2FA performs well in the benign setting when using *Len3* with One-Error (FRR = 3.38%), *Len4* with One-Error (FRR = 5.04%), and *Len5* with Two-Error (FRR = 4.90%) settings. With these settings, SG-2FA also performs well in the attack settings (low FARs) that we present next.

2) Attack Settings: To evaluate the security offered by SG-2FA against the impersonation attacks, similar to the benign setting, we employ the LOSO model. For each user's attack sample, we use positive instances from all the users while the negative instances were taken from other users. Below, we present the FAR results in different attack settings presented in Section II.

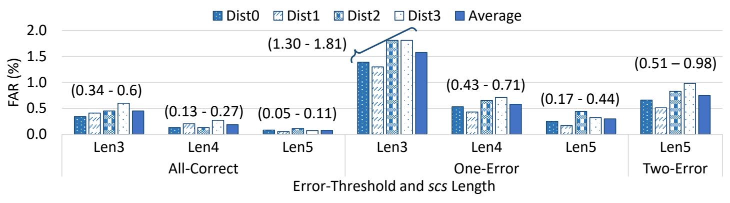
Threat 1 – Regular Wrist Movements: Appendix Figure 8a shows the FARs of SG-2FA for high-threat setting, where an attacker attempts to login when a user is performing his everyday regular activities. With the All-Correct setting, we achieved the average FARs$\leq$0.5%. The average FARs were still below 1.6% for all *scs* lengths when the *scs-Verifier* was made flexible by setting error-threshold to One-Error. With the Two-Error setting, we achieved the average FAR of 0.98% for *Len5*. We note that we achieved FAR of 0% with all error-threshold settings for the attack scenario where the user keeps his wrist static, the most likely occurring scenario during a random attack. These results show that the SG-2FA is robust against such attack settings where users are executing their regular activities.

Threat 2 – Text Typing: Similar to the high-threat setting, Appendix Figure 8b shows the FARs of SG-2FA for the medium-threat setting, where an attacker attempts to login when a user is typing regular text on his terminal. The FARs of SG-2FA in such attack settings are lower than for the attack settings when the user is executing other activities. The FARs were extremely low ($<$0.5%) for all the settings except for *Len3* with One-Error setting (around 3%) potentially because the gesture generated when typing on the keyboard may be significantly different from the seamless gestures used in SG-2FA. As can be seen from the figure that SG-2FA is 100% successful in detecting adversary in such attack scenarios for most of the settings. The FARs would be even lower when considering the regular terminal access activities rather than the continuous typing activities considered in our analysis. These results indicate that SG-2FA is well capable of identifying the adversary in such attack settings, where the user is typing regular texts.

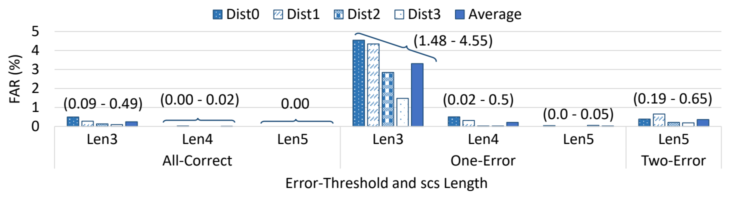
Threat 3 – Password Typing: Appendix Figure 8c shows the FARs of SG-2FA for low-threats setting, where an attacker attempts to login when the victim user is also trying to log into another remote server. Although it is named as low-threat setting, the underlying attack model represents a very strong model compared to the attack models considered previously. In such a threat setting, with All-Correct, we achieved average FARs of (3.91-11.06)%. When using the One-Error setting, the average FARs increased to (9.93-26.81)%. Further, with the Two-Error setting, we achieved average FAR of 19.70% for *Len5*. Although SG-2FA achieved high FARs in such an attack setting, we note that this attack model is extremely strong and rare to happen in practice. The time when an attacker tries to login overlaps with the time a user attempts to log in is a very strong assumption.

These results show that an attack against SG-2FA cannot succeed unless the timeframe during which the attacker supplies the password (or creates seamless gestures) overlaps with the timeframe during which the user types the password. However, such overlapping would be hard to achieve in practice (even with a traditional phishing attack) since the login activity is of short duration (merely a few seconds). Even with this overlapping, the highest success probability of the attack is $<$25% because the typing of *scs* may not overlap and/or their *scs* gestures may not completely synchronize. Further, even considering the rare chances that both the attacker and the victim type the password at the same time, the attack success rate would drop significantly, below 25%.

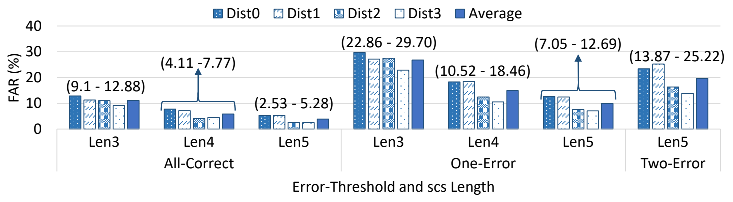
Summary of Results: As mentioned earlier, the low-threat setting represents the very strong attack model and is extremely rare to occurs. To evaluate the performance of SG-2FA well, we present the overall results in two parts – a) including



(a) High Likelihood Threat – Regular Wrist Movements



(b) Medium Likelihood Threat – Text Typing



(c) Low Likelihood Threat – Password Typing

Fig. 8: Attack Setting. False Acceptance Rate (FAR) of SG-2FA for various scs length and scs key-distance setting with three different error-threshold settings. "Average" bar on each scs length setting shows the average of FAR over all the key distances.

TABLE I: Summary. The performance – FRR, FAR, Precision, Recall, and F1-Score (in terms of %) – of SG-2FA for various error-thresholds and scs length settings while including/excluding the low likelihood attack setting, specifically Threat 3 – Password Typing. Highlighted cells show best parameter settings.

Err-Threshold	SCS Len	FRR	FAR	Prec.	Recall	F1-Score
Including Threat 3 – Password Typing (the low likelihood threat)						
One-Err.	*Len3*	2.86	4.71	97.27	95.40	95.97
	Len4	4.47	1.93	98.29	97.92	98.03
	Len5	7.20	1.19	98.54	98.44	98.47
Two-Err.	*Len5*	5.10	2.56	97.28	97.28	97.46
Excluding Threat 3 – Password Typing (the low likelihood threat)						
One-Err.	*Len3*	2.86	2.23	98.20	97.73	97.87
	Len4	4.47	0.45	99.30	99.29	99.29
	Len5	7.20	0.19	99.32	99.33	99.32
Two-Err.	*Len5*	5.10	0.60	99.11	99.09	99.10

the low-threat setting, and b) excluding the low-threat setting. Table I presents the overall performance of SG-2FA averaged over all the key-distances of scs. The first part shows the results when considering the low-threat setting and second part shows the results without considering it. Highlighted cells show the results of SG-2FA with the setting, where SG-2FA performs best in both the benign and the attack scenarios. The best performance of SG-2FA(in terms of low FAR with reasonable FRR) was achieved when using *Len4* with One-Error, and *Len5* with the Two-Error setting. When considering the low-threat, with *Len4* and One-Error, we achieved FRR of 4.47%, FAR of 1.93%, and F1-Score of 98.03%. With *Len5* and Two-Error, we achieved FRR of 5.10%, FAR of 2.56%, and F1-Score of 97.46%. The performance of SG-2FA improved when excluding the low-threat setting. With the best parameter setting, we achieved FARs$\leq$ 0.6% and F1-Score$>$ 99%. Further, for a random attacker, it is highly likely that when he launches the attack, the user will keep his hand static (e.g., on a desk, inside the pocket, etc.), where we achieved FAR of 0%.

These results show that SG-2FA (with the best parameter settings) performs well in both the benign and the attack settings. It can successfully identify the legitimate user and detect any fraudulent user with a high accuracy (>99%), particularly in high and medium threat. We note that the FRR/FAR of $\leq 5\%$ is considered reasonably good for the behavioral authentication systems [26], [49], [27]. Further, most of the existing authentication systems (e.g., Duo Push, Phone-call, Duo App Passcode, SMS passcode) have comparable (and even higher) error rate ($\geq 5\%$) to our approach, potentially stemming from misreading and mistyping [30]. Moreover, they require the user to spend quite high time (>10 seconds on average) during the 2FA ceremony compared to our approach (˜5 seconds). The performance of SG-2FA could further be improved with a wrist-device having a much higher sampling rate (e.g., 500 Hz vs. 200 Hz) for motion sensor recordings, such as the Shimmer bracelet [38].

Entropy of scs: As we can see from the table, with the One-Error setting, increasing the length of scs decreases the FAR, however, it also increases FRR at the same time. With longer scs length and proper selection of error-threshold (e.g., Two-

Error with *Len5*), we could achieve balanced FRR and FAR. Thus, with the increase of *scs* length (and a little increase in latency) and proper error-thresholdization, the overall performance (both the security and the usability) of SG-2FA can be further improved.

VIII. Other Related Work

Sound-Proof [22] is a minimal effort phone-based 2FA system that leverages ambient sounds to detect the proximity of the second factor (phone) and the browser. However, this approach is insecure against co-located attackers as well as remote attackers who can predict the user's ambient environment [41]. Moreover, this approach is also insecure against the remote attackers who can make the second factor device, i.e., the phone, create some predictable or already known sounds, as shown in [40]. Addressing these security vulnerabilities found in Sound-Proof, Shrestha et al. [41] and Liu et al. [25] introduced other sound-based low-effort 2FA schemes, called *Listening-Watch* and *Typing-Proof*, respectively. Listening-Watch is based on a wearable device and browser generated active speech sounds. The active sounds used in Listening-Watch may greatly impact the usability of the system. Further, to prevent proximity attacks, Listening-Watch relies on a rather low-quality of the smartwatch microphone, which may not be the case for future watches. In Typing-Proof, during the login process, the user needs to type a random code on the computer's keyboard and the login succeeds if the keystroke timing sequence from the browser matches with the recorded keystroke sound on the user's phone. Typing-Proof relies on the audio sound generated when the user types a random code using the keyboard. However, most of the keyboards, especially those found in the laptops, hardly generate any sounds. Further, Typing-Proof requires the users to keep their phones near the login terminal to record the keystrokes. Typing-Proof may not work well in the scenario where the user keeps the phone inside his pocket (or bag). In contrast, SG-2FA has better usability – mere typing of a short special sequence is sufficient to login, and yet can effectively defeat both the proximity and remote attackers.

Lewis et al. [24] proposed a *motion-based behavioral biometrics* mechanism to authenticate the user/wearer to her watch. In contrast, SG-2FA offers watch-based web 2FA and is not a biometric scheme. Further, SG-2FA is privacy-preserving and does not need a separate prediction model for each individual user – a generic model is sufficient.

Similar to SG-2FA, ZEBRA [26] and iAuth [23] utilize the wrist movements to authenticate the user. However, the overall design and detection algorithms of SG-2FA are completely different from those of ZEBRA and iAuth. To authenticate the user, ZEBRA compares the sequence of events it observed (e.g., typing, scrolling) with the sequence of events inferred using the wrist-motion readings. Unfortunately, unlike SG-2FA, ZEBRA is vulnerable to audio-visual based opportunistic attacks as shown in [21]. iAuth utilizes the phone-usage behavior of the user captured by the user's wrist-watch and the phone to authenticate the user. In contrast, SG-2FA authenticates the user based on the unique hand movement generated while typing a short sequence. Moreover, the authentication domain of these two schemes compared to SG-2FA is also completely different. ZEBRA and iAuth are targeted for the continuous authentication (de-authentication) for a local machine (e.g., shared space computer use) and a smartphone, respectively, while SG-2FA is targeted for the point-of-entry authentication to a remote web-server.

IX. Limitations and Future Work

In our study, a single desktop PC with an external keyboard was used to perform all logins and the watch was worn on the left hand of the user. Although we believe that the model/manufacturer of the keyboard, specifically the physical hardness, does not have much effect on SG-2FA, future work is needed to explore the impact of physical hardness on the performance of SG-2FA. SG-2FA was evaluated with only 30 volunteer participants, the majority of them being young with technical/CS background. A future study with a larger and diverse pool of users might be needed to generalize the scheme well. Like other related systems, SG-2FA may require users to undergo an initial one-time training session for familiarization purposes. In our study, participants were informed that wrist-motion data will be collected during the experiment, which might have influenced their typing behavior. Our study and evaluation also apply to a setting where a laptop is positioned on a desk given that the built-in keyboard on the laptop typically follow the same keyboard layout as an external keyboard used in our study. However, different orientation of the laptop, such as placing it on a lap, may impact the performance of SG-2FA. A rigorous future work is needed to explore more in this direction. Although we have evaluated SG-2FA against a *simulated phishing attack*, one future direction is to study the feasibility of the *real-time phishing-attack* against SG-2FA. Another interesting future direction is to experimentally evaluate the usability of SG-2FA vs. PIN-2FA and Tap-2FA.

X. Conclusion

In this paper, we presented SG-2FA, a strong and low-effort wearable (watch-based) 2FA scheme based on the notion of seamless typing gestures. SG-2FA carefully applies machine learning techniques to identify the unique wrist gesture produced while typing a short, fixed and non-memorized sequence of characters on the authentication terminal's browser. Unlike PIN-2FA, which requires opening the app on the watch and copying a random PIN from the watch to the browser, SG-2FA needs zero interaction with the watch and only a short sequence (displayed on the browser) to be typed that greatly improve usability over PIN-2FA. Security compared to Tap-2FA is also significantly improved since there is no reliance on the user's decision making, and the attacker cannot succeed to login unless the user's wrist is simultaneously undergoing the exact same seamless gestures. Notably, unlike behavioral biometric schemes, our approach is privacy-preserving as it requires no user-specific information or templates to be stored on the authentication server.

ACKNOWLEDGMENTS

The authors would like to thank anonymous reviewers for their feedback on the paper. This work is partially supported by National Science Foundation (NSF) under the grants: CNS-1547350, CNS-1526524, CNS-1714807, and CNS-1527795.

REFERENCES

[1] Y. AB., "Yubico — trust the net with yubikey strong two-factor authentication," https://www.yubico.com/, 2020, accessed: September 3, 2020.

[2] Apple Inc., "Apns overview," https://goo.gl/k37dLV, 2018, accessed: February 1, 2018.

[3] A. Bangor, P. Kortum, and J. Miller, "Determining what individual sus scores mean: Adding an adjective rating scale," *Journal of usability studies*, vol. 4, no. 3, pp. 114–123, 2009.

[4] 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 *IEEE Symposium on Security and Privacy*. IEEE, 2012, pp. 553–567.

[5] J. Brooke *et al.*, "Sus-a quick and dirty usability scale," *Usability evaluation in industry*, vol. 189, no. 194, pp. 4–7, 1996.

[6] Celestix, "Celestix hotpin two factor authentication," http://www.celestixworks.com/HOTPin.asp, 2020, accessed: September 3, 2020.

[7] S. Chiasson, P. C. van Oorschot, and R. Biddle, "A usability study and critique of two password managers." in *USENIX Security Symposium*, 2006, pp. 1–16.

[8] J. Colnago, S. Devlin, M. Oates, C. Swoopes, L. Bauer, L. Cranor, and N. Christin, "it's not actually that horrible: Exploring adoption of two-factor authentication at a university," in *Conference on Human Factors in Computing Systems*. ACM, 2018, p. 456.

[9] M. Conti, I. Zachia-Zlatea, and B. Crispo, "Mind how you answer me!: transparently authenticating the user of a smartphone when answering or placing a call," in *ACM Symposium on Information, Computer and Communications Security*. ACM, 2011, pp. 249–259.

[10] Duo Security Inc., "Duo mobile and apple watch," 2019, https://guide.duo.com/apple-watch.

[11] ——, "Easy authentication: Duo security," https://duo.com/solutions/features/user-experience/easy-authentication, 2020, accessed: September 13, 2020.

[12] J. Dutson, "User attitudes about duo two-factor authentication at byu," 2018.

[13] A. P. Felt, E. Ha, S. Egelman, A. Haney, E. Chin, and D. Wagner, "Android permissions: User attention, comprehension, and behavior," in *8th symposium on usable privacy and security*. ACM, 2012, p. 3.

[14] Gartner Inc., "Gartner says worldwide wearable device sales to grow 17 percent in 2017," https://goo.gl/z7DTz1, 2017.

[15] Google Inc., "Cloud messaging," https://developers.google.com/cloud-messaging/, 2019, accessed: February 1, 2018.

[16] ——, "Firebase cloud messaging — firebase," https://firebase.google.com/docs/cloud-messaging/, 2020, accessed: February 1, 2020.

[17] ——, "Google 2-step verification," https://www.google.com/landing/2step/, 2020, accessed: September 3, 2017.

[18] ——, "Sign in faster with 2-step verification phone prompts," 2020, Accessed: Sepetember 3, 2020. [Online]. Available: https://support.google.com/accounts/answer/7026266?co=GENIE.Platform%3DAndroid&hl=en

[19] C. Group, "Gac tizen client for google authenticator," 2016, Accessed; Last accessed 10 February, 2020. [Online]. Available: https://credelius.com/credelius/?p=120

[20] S. Grzonkowski, "Password recovery scam tricks users into handing over email account access," 2015, Accessed; Last accessed 28 October, 2018. [Online]. Available: https://goo.gl/x8QiZi

[21] O. Huhta, P. Shrestha, S. Udar, M. Juuti, N. Saxena, and N. Asokan, "Pitfalls in designing zero-effort deauthentication: Opportunistic human observation attacks," in *Network and Distributed System Security Symposium*, 2016.

[22] N. Karapanos, C. Marforio, C. Soriente, and S. Capkun, "Sound-proof: usable two-factor authentication based on ambient sound," in *USENIX Security Symposium*, 2015.

[23] W.-H. Lee and R. Lee, "Implicit sensor-based authentication of smartphone users with smartwatch," in *Proceedings of the Hardware and Architectural Support for Security and Privacy 2016*. ACM, 2016, p. 9.

[24] A. Lewis, Y. Li, and M. Xie, "Real time motion-based authentication for smartwatch," in *2016 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2016, pp. 380–381.

[25] X. Liu, Y. Li, and R. H. Deng, "Typing-proof: Usable, secure and low-cost two-factor authentication based on keystroke timings," in *Proceedings of the 34th Annual Computer Security Applications Conference*. ACM, 2018, pp. 53–65.

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

[27] M. Mohamed and N. Saxena, "Gametrics: towards attack-resilient behavioral authentication with simple cognitive games," in *Proceedings of the 32nd Annual Conference on Computer Security Applications*. ACM, 2016, pp. 277–288.

[28] S. RASCHKA, "Minmax scaling - mlxtend," https://rasbt.github.io/mlxtend/user_guide/preprocessing/minmax_scaling/, 2018, accessed: February 3, 2018.

[29] N. Ravi, N. Dandekar, P. Mysore, and M. L. Littman, "Activity recognition from accelerometer data," in *Aaai*, vol. 5, 2005, pp. 1541–1546.

[30] J. Reynolds, N. Samarin, J. Barnes, T. Judd, J. Mason, M. Bailey, and S. Egelman, "Empirical measurement of systemic 2fa usability," in *USENIX Security Symposium*, 2020, pp. 127–143.

[31] B. Ross, C. Jackson, N. Miyake, D. Boneh, and J. C. Mitchell, "Stronger password authentication using browser extensions." in *USENIX Security Symposium*. Baltimore, MD, USA, 2005, pp. 17–32.

[32] RSA Security, "Rsa securid hardware tokens — two factor authentication," 2018, accessed: September 3, 2020. [Online]. Available: https://goo.gl/eSjrFa

[33] S. Ruoti, J. Andersen, D. Zappala, and K. Seamons, "Why johnny still, still can't encrypt: Evaluating the usability of a modern pgp client," *arXiv preprint arXiv:1510.08555*, 2015.

[34] SAASPASS, "Unlock your computer and websites with your android wear smartwatch," 2016, Accessed; Last accessed 10 February, 2020. [Online]. Available: https://bit.ly/2SzMN2b

[35] SAFEPASSWD, "Password generator for a strong secure memorable password," 2018, Accessed: March 05, 2018. [Online]. Available: https://www.safepasswd.com/

[36] B. Schoon, "google prompt two-factor authentication now works on android wear devices for some," 2016, Accessed; Last accessed 10 February, 2020. [Online]. Available: https://9to5google.com/2016/11/19/google-prompt-android-wear/

[37] M. Shahzad, A. X. Liu, and A. Samuel, "Secure unlocking of mobile touch screen devices by simple gestures: you can see it but you can not do it," in *Proceedings of the 19th annual international conference on Mobile computing & networking*. ACM, 2013, pp. 39–50.

[38] Shimmer Inc., "Wearable sensor technology — wireless imu — ecg — emg — gsr," 2020, http://www.shimmersensing.com/.

[39] M. Shirvanian, S. Jareckiy, H. Krawczykz, and N. Saxena, "Sphinx: A password store that perfectly hides passwords from itself," in *Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on*. IEEE, 2017, pp. 1094–1104.

[40] B. Shrestha, M. Shirvanian, P. Shrestha, and N. Saxena, "The sounds of the phones: dangers of zero-effort second factor login based on ambient audio," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 908–919.

[41] P. Shrestha, , and N. Saxena, "Listening watch: Wearable two-factor authentication using speech signals resilient to near-far attacks," in *Proceedings of the 11th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. ACM, 2018.

[42] P. Shrestha and N. Saxena, "Hacksaw: biometric-free non-stop web authentication in an emerging world of wearables," in *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2020, pp. 13–24.

[43] Sony, "Sony smartwatch 3 swr50," https://goo.gl/touP3z, 2018, accessed: February 3, 2018.

[44] J. Sunshine, S. Egelman, H. Almuhimedi, N. Atri, and L. F. Cranor, "Crying wolf: An empirical study of ssl warning effectiveness." in *USENIX security symposium*, 2009, pp. 399–416.

[45] E. Ulqinaku, D. Lain, and S. Capkun, "2fa-pp: 2nd factor phishing prevention," in *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*, 2019, pp. 60–70.

[46] VDA Labs, "Phishing users using evilginx and bypassing 2fa," 2019, https://vdalabs.com/2019/10/01/phishing-users-using-evilginx-and-bypassing-2fa/.

[47] Weka, "Spreadsubsample," https://goo.gl/VGaAw6, 2018, accessed: February 3, 2018.

[48] H. Xu, Y. Zhou, and M. R. Lyu, "Towards continuous and passive authentication via touch biometrics: An experimental study on smartphones," in *Symposium On Usable Privacy and Security, SOUPS*, vol. 14, 2014, pp. 187–198.

[49] J. Yang, Y. Li, and M. Xie, "Motionauth: Motion-based authentication for wrist worn smart devices," in *Pervasive Computing and Communication Workshops (PerCom Workshops), 2015 IEEE International Conference on*. IEEE, 2015, pp. 550–555.

[50] Yubico, "U2f - fido universal 2nd factor authentication — yubico," https://www.yubico.com/solutions/fido-u2f/, 2020, accessed: September 3, 2020.

Appendix

A. *scs* Gyroscope Signal

Figure 9 shows the gyroscope signal of user's wrist when he types the *scs*– "#$#$".

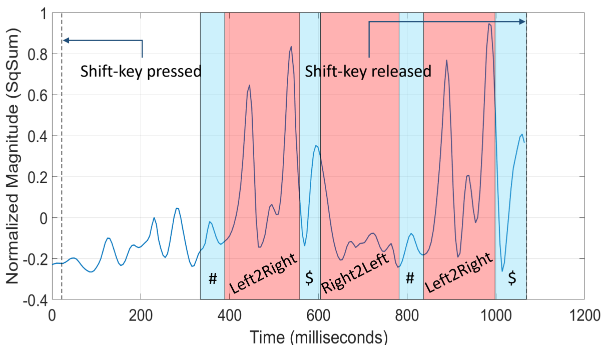


Fig. 9: Gyroscope measurements of user's wrist when he types scs– "#$#$". Left2Right – left-to-right; Right2Left – right-to-left wrist movement.

B. Feature List and Classifiers' Performance

We used 18 different features over each of the x, y, and z axis readings and from $SqSum$ of accelerometer and gyroscope sensors to classify a *scs-motion*, which are presented in Table II.

TABLE II: List of features used in SG-2FA.

Feature	**Description**
Minimum	minimum value in signal
Maximum	maximum value in signal
Mean	mean value of signal
Median	median value of signal
Variance	variance of signal
Standard deviation	standard deviation of signal
MAD	median absolute deviation
IQR	inter-quartile range
Power	power of signal
Energy	energy of signal
Spectral Entropy	distribution of energy in signal
Autocorrelation	similarity of signal
Kurtosis	peakedness of signal
Skewness	asymmetry of signal
Median frequency	median normalized frequency
Peak counts	average number of peaks and troughs per 100ms in signal
Peak-to-peak	peak-to-peak amplitude
Peak-magnitude-to-rms ratio	ratio of largest absolute value to root-mean-square (RMS) value of signal

As a classifier for *scs-Verifier* in SG-2FA, we tested several state-of-the-art machine learning algorithms. To measure the performance of each classifier, we employed 10-fold cross-validation approach, and averaged the F-Measure over three key distances of *scs*. Table III shows the classification performance of the various classifiers that we tested.

C. Demographics and Wrist Device Usage

Prior to the experiment, we asked the participants about their general demographics and wrist device usage questions. Specifically, we asked participants if they own any wrist-worn devices (e.g., smartwatch, regular watch, bracelet, or bands), how often they use it, and in which hand they usually wear it. The demographics of the participants and their wrist device usage are summarized in Table IV.

TABLE III: Classification performance (F-Measure averaged over three key distances of scs) of various classifiers based on 10-fold cross-validation approach for three different scs lengths. Note that the presented performance are for predicting scs-motion (not to predict scs entry). Highlighted row shows the performance of classifier that outperformed others.

Classifier	F-Measure (%)		
	SCS-Len3	SCS-Len4	SCS-Len5
Logistic	96.03	96.89	97.13
Simple Logistic	96.54	97.15	96.92
Multilayer Perceptron	95.39	95.50	95.85
NaiveBayes	74.77	75.85	76.85
RandomForest	97.40	97.89	97.83
RandomTree	94.41	95.27	95.47
SVM	96.11	96.67	96.24

TABLE IV: Demographics and wrist device (e.g., smart or regular watch, bracelet, bands, etc.) usage of participants (N = 30). 'CS' represent 'Computer Science'.

(a) Demographics

Category	# of subjects
Gender	
Male	22
Female	8
Age	
<25	2
25-35	26
>35	2
Field	
CS	26
Non-CS	4
Handedness	
Right	28
Left	2

(b) Wrist Device Usage

	# of subjects	
Wrist device	Smart	Normal
Own wrist device		
Yes	11	21
No	19	9
Regularity of usage		
Frequent	5	15
Sometime	6	6
Worn on		
Left Hand	10	19
Right Hand	1	0
Both Hands	0	2

D. List of *scs* and Typing Time of *scs* & *sg_pwd*.

Table V shows the list of *scs* chosen by the participants during the experiment and Table VI shows the average time to type *scs* and *sg_pwd* by the participants.

TABLE V: List of SCS chosen by the participants.

SCS key-distance	**SCS Chosen by Users**
SCS Length = 3 (Len3)	
Dist0	!@!, #$#, $%$, $#$, ˜!˜, %ˆ%, @#@
Dist1	#%#, %#%, $ˆ$, @$@, ˜@˜, !#!
Dist2	˜#˜, #ˆ#, %@%, @%@, !$!
Dist3	@ˆ@, ˜$˜, %!%, !%!
SCS Length = 4 (Len4)	
Dist0	%ˆ%ˆ, !@!@, $%$%, #$#$, @#@#, $#$#, ˜!˜!, @!@!
Dist1	#%#%, @$@$, $ˆ$ˆ, !#!#, #!#!, ˜@˜@
Dist2	˜#˜#, @%@%, #ˆ#ˆ, !$!$, $!$!
Dist3	@ˆ@ˆ, ˜$˜$, !%!%, %!%!, ˆ@ˆ@
SCS Length = 5 (Len5)	
Dist0	%ˆ%ˆ%, ˜!˜!˜, @!@!@, @#@#@, !@!@!, $#$#$, $%$%$, #$#$#
Dist1	#%#%#, ˜@˜@˜, $@$@$, $ˆ$ˆ$, #!#!#, !#!#!, ˆ$ˆ$ˆ, @$@$@
Dist2	$!$!$, @%@%@, ˜#˜#˜, #ˆ#ˆ#, %@%@%, !$!$!
Dist3	ˆ@ˆ@ˆ, ˜$˜$˜, @ˆ@ˆ@, %!%!%, !%!%!

TABLE VI: Average (standard deviation) time taken by the participants to type scs and sg_pwd.

scs Length	scs (sec)	sg_pwd (sec)
Len3	1.21 (0.22)	5.13 (1.38)
Len4	1.51 (0.31)	5.18 (1.35)
Len5	1.89 (0.38)	5.49 (1.35)

E. SG-2FA vs. Other Schemes

We used the framework of Bonneau et al. [4] to analytically compare SG-2FA with well-known browser compatible TFA schemes, namely Google two-step Verification (Google 2SV), and Sound-Proof. The framework of Bonneau et al. considers several parameters, termed as "benefits", derived from the perspective of usability, deployability, and security that an authentication scheme should ideally provide. The summary of the overall comparison using the framework of Bonneau et al. is shown in Table VII

*TABLE VII: Comparing SG-2FA against PIN-2FA and Tap-2FA using the framework of Bonneau et al. [4]. '$*$' represents that the scheme "**offers**" the benefit, '$+$' represents that the scheme "**somewhat offer**" the benefit, and '$-$' indicates that the scheme "**does not offer**" the benefit.*

	Usability								**Deployability**						**Security**										
Scheme	*Memorywise-Effortless*	*Scalable-for-Users*	*Nothing-to-Carry*	*Physically Effortless*	*Easy-to-Learn*	*Efficient-to-Use*	*Infrequent-Errors*	*Easy-Recovery-from-Loss*	*Accessible*	*Negligible-Cost-per-User*	*Server-Compatible*	*Browser-Compatible*	*Mature*	*Non-Proprietary*	*Resilient-to-Physical-Observation*	*Resilient-to-Targeted-Impersonation*	*Resilient-to-Throttled-Guessing*	*Resilient-to-Unthrottled-Guessing*	*Resilient-to-Internal-Observation*	*Resilient-to-Leaks-from-Other-Verifiers*	*Resilient-to-Phishing*	*Resilient-to-Theft*	*No-Trusted-Third-Party*	*Requiring-Explicit-Consent*	*Unlinkable*
PIN-2FA	$-$	$-$	$+$	$-$	$*$	$+$	$+$	$+$	$+$	$*$	$-$	$*$	$*$	$*$	$+$	$+$	$*$	$*$	$-$	$*$	$+$	$*$	$*$	$*$	$-$
Tap-2FA	$-$	$-$	$+$	$-$	$*$	$+$	$+$	$+$	$+$	$*$	$-$	$*$	$*$	$*$	$+$	$+$	$*$	$*$	$-$	$*$	$+$	$*$	$*$	$*$	$-$
SG-2FA	$-$	$-$	$+$	$-$	$*$	$*$	$+$	$+$	$*$	$+$	$-$	$*$	$-$	$*$	$+$	$+$	$*$	$*$	$-$	$*$	$*$	$*$	$*$	$*$	$-$