NomadiKey: User Authentication for Smart Devices based on Nomadic Keys

Artur Souza[1], **Ítalo Cunha**[2], **Leonardo B. Oliveira**[2]

[1] Student – Computer Science Department – Universidade Federal de Minas Gerais

[2]Advisors – Computer Science Department – Universidade Federal de Minas Gerais

Abstract. *The growing importance of smart devices calls for effective user authentication mechanisms. In this paper, we argue that state-of-the-art authentication mechanisms are either vulnerable to known attacks or do not fully meet usability needs. To address this problem we design NomadiKey, a user-to-device authentication mechanism based on nomadic keyboard keys. NomadiKey increases security level by placing keys at different screen coordinates each time it is activated. Besides, NomadiKey preserves usability by maintaining the traditional relative position of keys. To increase security even further, we also design an extension to NomadiKey, called NomadiKey++, that employs out-of-band channels to protect the user from shoulder-surfing attacks.*

1. Introduction

The deployment of denser smart environments and increasing user dependence on their functionalities impose more stringent requirements on user security and privacy. Smart devices can store gigabytes of personal, private, and potentially sensitive, user data. To protect these data, devices often rely on user authentication mechanisms. These mechanisms can be divided into three categories: (i) something the user has, (ii) something the user is, and (iii) something the user knows.

Users can authenticate using a physical authenticator they have, like a token or smart card [Bojinov and Boneh 2011]. However, in most cases, an adversary can simply simultaneously steal the smart device and the authenticator and access user data.

Users can authenticate through their own biological features like fingerprints, voice, or face. Some previous works propose and evaluate biometric authentication factors from unique body characteristics, like hand characteristics [Pan et al. 2013], iris recognition [Mock et al. 2012], and ECG measurements [Rostami et al. 2013]. Other previous works consider behavioral patterns to authenticate the user, often transparently [Wang et al. 2015]. Biometric schemes usually require simple actions from users during authentication and are vulnerable to few attacks. These schemes, however, require user's biometric features to be stored as secrets in the device, which raises privacy concerns.

In most cases, a user authenticates onto a smart device inputting a secret previously configured on the device. This type of authentication is the most common on smart devices [Egelman et al. 2014]. The security and usability of these mechanisms are heavily affected by the secret chosen; complex secrets increase security but impair usability [Dell'Amico et al. 2010]. Previous work, similar to ours, try to achieve a better trade-off between security and usability by, for instance, motivating users to create stronger but easy-to-input passwords [Haque et al. 2013], using short strokes instead of

Table 1. Worst-case scenario of information available to adversaries

	TOUCH INFORMATION		
ATTACK	Location	Order	Content
Smudge	●		
Vision	●	●	
Shoulder-surfing	●	●	●

simply touching buttons [Arif and Mazalek 2013], or designing extensions to resist specific attacks [Yue et al. 2014].

We consider three classes of attacks against authentication mechanisms, namely *smudge*, *computer vision*, and *shoulder-surfing* attacks. *Smudge attacks* use digital image processing to identify where the screen was touched from residues on the screen (e.g., oil or dust) [Aviv et al. 2010]. *Computer vision attacks* use computer vision techniques to estimate the position of the fingers (and shadows) as they approach the screen to infer when and where the screen was touched [Yue et al. 2014]. In *Shoulder surfing attacks*, adversaries observe the user authenticating, or videos of the user authenticating, to obtain touch position, order, and screen contents. Evaluation of these attacks shows success rates as high as 92%, 91%, and 95% respectively. Our adversary models are summarized in Table 1.

To defend against the aforementioned attacks, we designed an authentication mechanism robust to these attacks while incurring a negligible increase in authentication speed (Section 2). NomadiKey [Cotta et al. 2016], as it is called, places keys at random *absolute* positions on the screen each time it is activated, preventing adversaries from inferring which keys are pressed during authentication and improving security. Besides, NomadiKey preserves usability by maintaining keys' *relative* positions, helping users navigate the keyboard and locate keys. To protect users against shoulder-surfing attacks, we also present a security extension for NomadiKey that employs vibrations as an out-of-band channel during authentication. The out-of-band channel prevent observing adversaries from retrieving the entire authentication secret through a shoulder-surfing attack.

We compare NomadiKey and NomadiKey++ with five other authentication schemes in face of smudge, vision, and shoulder surfing attacks (Section 3). We first present a worst-case model of each attack, then use statistical modeling to quantify security of each authentication mechanism under no attack and under each of the attacks. We also compare the usability of NomadiKey and NomadiKey++ to the usability of classic PIN authentication and of PIN authentication on random keyboards (Section 4).

In this paper, we make the following novel contributions:

1. NomadiKey++, A security extension to NomadiKey to protect users against shoulder-surfing attacks.
2. Expanded analytical evaluation of NomadiKey, including other authentication mechanisms and shoulder-surfing attacks.
3. Analytical and empirical evaluation of NomadiKey++, comparing it to NomadiKey and other existing authentication mechanisms.

This paper is an extension of previous work. Former versions of NomadiKey appeared on SBSeg [Neto et al. 2015] and IEEE ICC [Cotta et al. 2016] (The student is a coauthor, under the name Artur Luis Fernandes). The student, together with his advisors, were the sole responsible for conceiving, designing, implementing and evaluating the security extension presented in this paper. The student played a key role in all stages of the development of the original version of NomadiKey as well and was responsible for presenting the work at ICC in Kuala Lumpur, Malaysia. At that time, he also gave an interview about the work, which appeared in online venues like IEEE Spectrum[1] and Android Community[2]. Finally, NomadiKey and NomadiKey++ were the subject of the student's final graduation project at UFMG. These and other information about NomadiKey can be found online at its page[3].

2. NomadiKey

In what follows, we present NomadiKey, a new authentication mechanism for smart devices that targets what we believe is a good balance between security and usability. The essence of NomadiKey is an algorithm that allocates positions for keys on the screen as freely as possible but under the constraint that the relative order among them is the same as in a traditional keyboard. More precisely, NomadiKey places keys at random *absolute* positions while constraining the *relative* position of keys. Figure 1 shows an example keyboard built by NomadiKey. Keys are in random positions, but observe that keys on the first line (1, 2, and 3) are above other keys and keys on the first column (1, 4, and 7) are to the left of other keys.

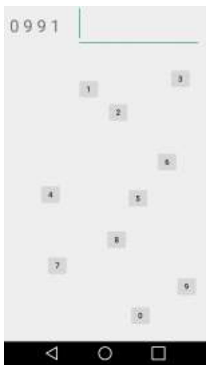

Figure 1. NomadiKey.

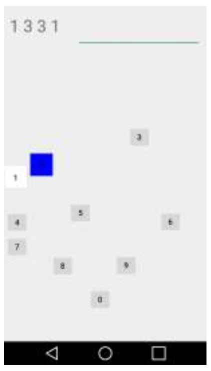

Figure 2. NomadiKey++.

2.1. NomadiKey: Key Position Algorithm

Let S_w and S_h be the screen width and screen height, respectively; and let B_w and B_h be the button width and button height, respectively; let C and R be the number of columns and rows in the keyboard, respectively. We consider the top-left corner of the screen is the origin, i.e., the point $(0, 0)$. We show pseudocode for NomadiKey in Algorithm 1.

NomadiKey partitions the screen into columns and randomizes the coordinate $x(c)$ where each column $c \in [1, C]$ starts. For each column c, NomadiKey computes its minimum possible starting coordinate $x_{\min}(c)$ as $x_{\min}(c) = x(c-1) + \alpha B_w$. The αB_w term

[1]http://spectrum.ieee.org/tech-talk/telecom/security/this-mobile-security-feature-will-annoy-you-but-it-will-also-protect-your-phone

[2]https://androidcommunity.com/nomadikey-shrinks-passcode-entries-to-prevent-smudge-attacks-20160603/

[3]https://sites.google.com/view/nomadikey/

ensures that there is space for placing keys in column $c-1$, where $\alpha > 1$ is a factor that defines the minimum width of a column as a function of the button width B_w. We set $x(c-1) = x_{\min}(c-1)$ if column $c-1$ has not been placed yet. This makes $x_{\min}(c)$ the smallest coordinate that still reserves at least αB_w for each unplaced column to the left of column c. We define $x(0) = -\alpha B_w$. Similarly, the maximum possible starting coordinate $x_{\max}(c)$ for column c is $x_{\max}(c) = x(c+1) - \alpha B_w$. To cover the case where $c = C$, we define $x(C+1) = S_w + \alpha B_w$.

NomadiKey then chooses coordinate $x(c)$ where column c starts uniformly distributed between $x_{\min}(c)$ and $x_{\max}(c)$. NomadiKey repeats this computation exchanging x for y, C for R, and B_w for B_h to compute $y_{\min}(r)$ and $y_{\max}(r)$ for each row $r \in [1, R]$ (see Algorithm 1). NomadiKey then chooses the coordinate $y(r)$ where each row r starts uniformly distributed between $y_{\min}(r)$ and $y_{\max}(r)$.

The algorithm ensures each grid cell is at least αB_w units wide and at least αB_h units high, which guarantees NomadiKey can place all keys. NomadiKey chooses the x coordinates of keys on column c uniformly distributed between $x(c)$ and $x(c+1)$; similarly for y coordinates.

Procedure 1 Key Placement

```
 1: procedure COMPUTE $x_{\min}(c)$
 2:     if column $c-1$ has been placed then
 3:         return $x(c-1) + \alpha B_w$
 4:     else
 5:         return $x_{\min}(c-1) + \alpha B_w$
 6: procedure COMPUTE $y_{\max}(r)$
 7:     if row $r+1$ has been placed then
 8:         return $y(r+1) - \alpha B_w$
 9:     else
10:         return $y_{\max}(r+1) - \alpha B_w$
    {Procedures for $x_{\max}$ and $y_{\min}$ are analogous}
11: procedure NOMADIKEY
12:     for $c \in$ random$(1, C)$ do
13:         $x(c) \leftarrow U(x_{\min}(c), x_{\max}(c))$
14:     for $r \in$ random$(1, R)$ do
15:         $y(r) \leftarrow U(y_{\min}(r), y_{\max}(r))$
16:     for $c \in [1, C]$ do
17:         for $r \in [1, R]$ do
18:             place key in column $c$, row $r$ at position
                $U(x(c), x(c+1)),\quad U(y(r), y(r+1))$
```

2.2. NomadiKey++: Shoulder-Surfing Protection

By randomizing the keys' absolute position, NomadiKey strikes a higher security level against smudge and vision attacks (Section 3). However, NomadiKey remains vulnerable to shoulder-surfing adversaries, that can identify where each key is placed. To protect the user against this type of attack, we have designed an extension to NomadiKey based on out-of-band channels.

Adversaries perform shoulder-surfing attacks by positioning themselves near unsuspecting users and observing them authenticate. To protect against this type of attack, we designed an extension to NomadiKey, called NomadiKey++, that employs vibrations as an out-of-band channel during authentication. Because adversaries are unable to detect the vibrations by observing the user authenticate, they are no longer able to discover the entire authentication secret through shoulder-surfing attacks.

NomadiKey++ works as follows. When it is triggered, the user is presented with a NomadiKey keyboard, as described in Section 2.1. As soon as the keyboard appears, NomadiKey++ starts highlighting keys one by one. In two random digits, the device vibrates while highlighting the key. After all keys have been highlighted, the user authenticates himself by entering his secret and the two keys indicated by the vibrations. The extra keys can be included in any order and position, even between digits of the secret. Figure 2 shows an example of NomadiKey++ highlighting the key 2.

3. Analytical Evaluation

In this section, we compare the security of NomadiKey and NomadiKey++ with classic and random PIN keyboard layouts, pattern-based authentication, LG's new Knock Code[4], and Samsung's Digital Door Lock[5]. We compare all these authentication mechanisms through their *security coefficient*, their number of possible distinct authentication secrets. We compare the security coefficients under safe operation ("no attack") as well as under smudge, vision, and shoulder-surfing attacks, as defined in our attack models in Section 1.

Table 2 shows closed formulas for the security coefficient of the evaluated mechanisms under safe operation and under each attack. We express security level as a function of the secret *length* n, the number of distinct keys d, the repetitions r_i of a key i and the number of random digits f added by NomadiKey++ and Digital Lock. We consider as keys the digits in PINs, NomadiKey and Digital Lock; the knocks in Knock Code; and dots for pattern authentication. Note we consider a more generic version of Digital Lock that adds f random digits instead of 2.

Security under safe operation. Under safe operation, the security coefficient of keyboard authentication grows exponentially with secret *length*. We note Knock Code's security level is equivalent to that of a $2{\times}2$ keyboard. The exponent's base is the number of possibilities for each PIN digit; or screen quadrants for knocking. The security of pattern-based authentication, in turn, depends on the number of dots available for pattern continuation, which is given by the n-permutation of the nine possible dots. Note that the security coefficient for pattern-based authentication is lesser or equal to the n-permutation of 9 dots. This is because not all patterns are possible for every dot subset. Also note that the extra digits added by Digital Lock and NomadiKey++ do not increase security under safe operation because an adversary has to guess only the n original PIN digits.

Security under smudge attacks. Smudge attacks allow adversaries to identify the location where the screen was touched, but not the order. Because of their predictable layout, classic PIN, pattern, and Knock Code authentication are vulnerable to smudge attacks. For classic PIN authentication, each touch allows the adversary to identify one number in the PIN. After identifying the numbers in the PIN, the adversary needs to guess the order in which they should be entered. Again, Knock Code's security is equivalent to that of a $2{\times}2$ keyboard. For pattern-based authentication, we note each dot can only be visited

[4]http://www.lg.com/us/mobile-phones/knockcode
[5]http://www.samsungdigitallife.com/DigitalDoorLock.php

Table 2. Security coefficient under different attack models.

Mechanism	Security coefficient			
	Safe Operation	Smudge attack	Vision attack	Shoulder Surfing
Classic PIN	10^n	$\frac{n!}{\prod_{i=1}^{d} r_i!}$	1	1
Random Keyboards	10^n	$\binom{10}{d}\frac{n!}{\prod_{i=1}^{d} r_i!}$	$\frac{10!}{(10-d)!}$	1
NomadiKey	10^n	$P\frac{n!}{\prod_{i=1}^{d} r_i!}$	P	1
Pattern	$\leq \frac{9!}{(9-n)!}$	2	1	1
Knock Code	4^n	$\frac{n!}{\prod_{i=1}^{d} r_i!}$	1	1
Digital Lock	10^n	$G_f^{n+f}\frac{n!}{\prod_{i=1}^{d} r_i!}$	1	1
NomadiKey++	10^n	$G_f^{n+f} * P\frac{n!}{\prod_{i=1}^{d} r_i!}$	$G_f^{n+f} * P$	G_f^{n+f}

$n =$ secret length $d =$ distinct keys $r_i =$ repetitions of key i $f =$ number of random digits added

once, i.e. the pattern is a Hamiltonian path. An adversary can authenticate by inputting the pattern seen in the smudge in forward and reverse directions.

For PIN authentication on random keyboards, knowing where the user touched the screen does not provide any information on which keys were pressed. The only information revealed is the number of distinct digits in the secret. For any d distinct touch points, the adversary needs to try all $\binom{10}{d}$ key combinations. For each key combination, the adversary has to try all possible orderings, as in classic PIN authentication.

NomadiKey is a middle-ground between PIN authentication on classic and random keyboards. The random absolute position of NomadiKey's keys prevents an adversary from discovering exactly which keys were pressed based on smudge touches. As in random keyboards, the adversary must try a number of key combinations P and all possible orderings for each key combination. The number of key combinations P, however, is less than $\binom{10}{d}$ because the adversary can ignore key combinations that contradict NomadiKey's restriction on keeping the relative position of keys. For instance, if the smudge reveals two perfectly aligned touches in a vertical line, the adversary can infer that the keys belong to one column and prune the set of possible values for each key.

In Digital Lock and NomadiKey++, when analyzing the smudge, an adversary is unable to distinguish the digits belonging to the PIN from the f random digits added by the authentication mechanism. In order to discover the user's secret, the adversary must first discover which digits belong to the user's PIN. The number of possibilities for the adversary to remove the random digits from the user's PIN is given by the number of distinct combinations of f digits that can be removed from the $n + f$ digit set. Using generating functions, the number of combinations can be modeled as a product of equations of the form $(x^0 + x^1 + ... + x^{r_i})$. The number of combinations is given by the coefficient of the x^f component in the expanded form of the product. In our evaluation, we use G_f^{n+f} to express the number of possibilities of choosing f digits from an $n + f$ set.

Finally, the security coefficient under *smudge* attacks of Digital Lock is given by the number of combinations G_f^{n+f} and the number of permutations of n digits. That is, the adversary has to, first, discover which are the real digits from the user's PIN and, second, discover the real order of the digits of the real PIN. For NomadiKey++, the security coefficient is given by G_f^{n+f}, the permutations of n, and the parameter P. That is, the adversary has to test all the permutations of all the G_f^{n+f} combinations of all the $n + f$ sets that do not contradict NomadiKey's restrictions, based on the touches revealed by the *smudge* attack.

Security under vision attacks. Vision attacks allow an adversary to know where the screen was touched, and in which order. Classic PIN authentication, pattern-based authentication, Knock Code, and Digital Lock have "static keys", so vision attacks give full information over which keys were pressed and in which order. Besides, Digital Lock's random digits are always entered before the user's real PIN digits, allowing an adversary to know which digits do not belong to the user PIN. Hence, the security coefficient of these mechanisms under vision attacks is 1.

NomadiKey and PIN on random keyboards provide some security against vision attacks as an adversary does not know exactly which keys were pressed. In NomadiKey, the adversary has to try all possible P key combination for a given set of touch points; similarly, an adversary has to try n-permutations of 10 keys for PIN authentication on random keyboards.

Unlike Digital Lock, the random digits of NomadiKey++ can be added at any point during authentication, meaning an adversary cannot distinguish random digits from real PIN digits based on the touch position and order. In NomadiKey++, the adversary still has to decide which digits belong to the user's secret for all possible keyboard configurations of NomadiKey.

Security under shoulder-surfing attacks. Our model for shoulder-surfing attacks allows an adversary to identify the exact keys touched in NomadiKey and random keyboards, completely revealing the authentication secret. NomadiKey++ is the only mechanism that provides a non-trivial security coefficient. NomadiKey++ uses vibrations during authentication to communicate random extra digits to the user, which are entered together with the real PIN during authentication. A shoulder-surfing adversary cannot distinguish between real and random PIN digits. Because of this, the security coefficient for NomadiKey++ under shoulder-surfing remains G_f^{n+f}.

4. Empirical evaluation

In this section, we present results of our empirical evaluation of NomadiKey and NomadiKey++. We complete our security analysis by assessing the value of the parameter P and we compare the usability of NomadiKey, NomadiKey++, classic PIN keyboards, and random PIN keyboards using prototypes we developed for each mechanism.

4.1. Security Evaluation

To complete our security evaluation of NomadiKey and NomadiKey++, we evaluate the values of P empirically. Recall that P is the number of possible key combinations given a set of touch positions. To evaluate P, we generate up to 10^5 distinct random secrets out of the $10!/(10-d)!$ possible secrets with d distinct keys, for d varying from 1 to 7. We then generate one hundred different keyboard layouts for each secret using Algorithm 1 with $\alpha = 2$, ensuring each row (column) is at least two buttons wide (high). Finally, we simulate a computer vision attack on each of the 10^7 (secret, layout) pairs to estimate the distribution of P.

Figure 3 shows the distribution of P. We observe that, for PINs with 4 distinct keys, the security coefficient of NomadiKey and NomadiKey++ is increased by more than 50 times in the median case. Vertical trends for $d = 1$ and $d = 2$ happen for touch positions where an adversary would have to try all possible d-key combinations, as for PIN authentication on random keyboards.

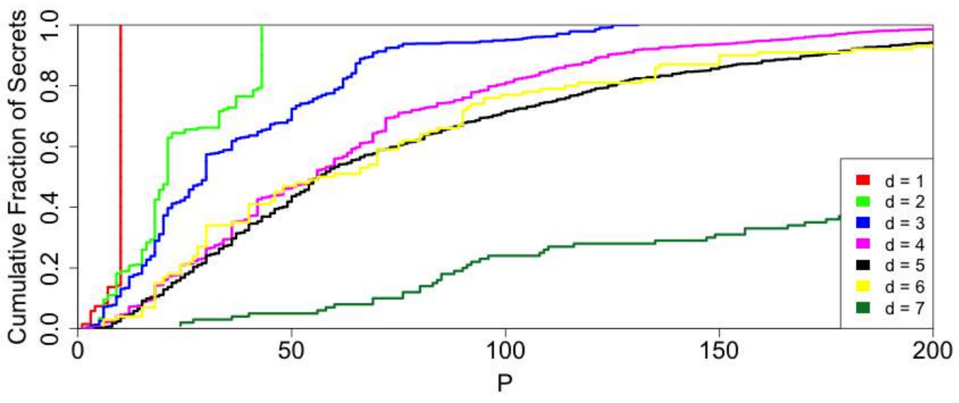

Figure 3. Empirical evaluation of the number P of possible key combinations in NomadiKey for a given set of touch positions.

4.2. Usability evaluation

To evaluate the usability of NomadiKey and NomadiKey++, we compare them to PIN authentication on classic and random keyboards. We measure usability as the *authentication delay*, how long it takes for users to authenticate. We implemented the four authentication mechanisms in an Android application. We considered PINs of four digits, based on average secret length of 4.5 digits found in previous work [Harbach et al. 2014].

We asked volunteers to authenticate in our developed application using each of the four authentication mechanisms. For each mechanism, the user was presented with five different random secrets. The user authenticated three times with each of the five secrets for a total of 15 authentications per mechanism. The tests were performed with 20 volunteers, 10 male and 10 female, using two devices: an LG G3 and an LG G4, both devices have screens of 5.5”.

Figure 4 shows authentication delay for NomadiKey. As expected, authentication delay for NomadiKey is a middle ground between traditional and random PIN authentication. We note that authentication times for NomadiKey are closer to those of random PINs than classic PINs. This is because, besides its nomadic nature, keys on NomadiKey are smaller than those used in other keyboards. Having smaller keys allows NomadiKey to increase security, as keys can be better spread on the screen, but it makes it harder for users to reach and press keys, decreasing usability.

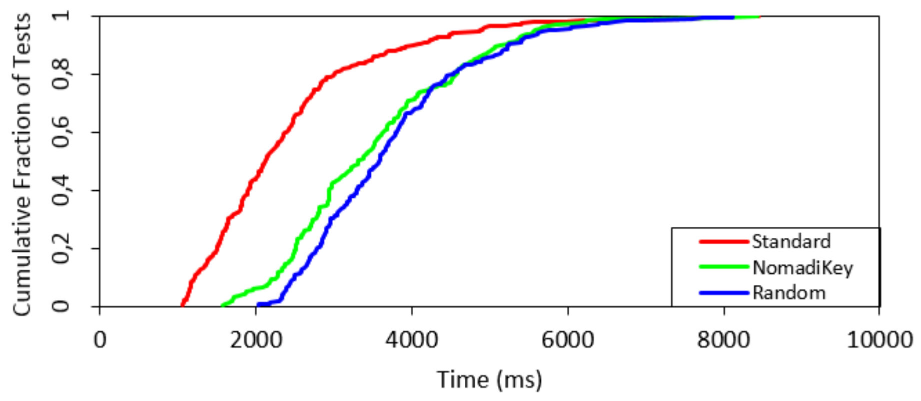

Figure 4. Usability of NomadiKey and PIN authentication.

Figure 5 shows results for NomadiKey++. It can be noted that authentication delays on NomadiKey++ are much higher than authentication delays on other authentication mechanisms (Figure 4). There are mainly two reasons for this. First, PINs entered for NomadiKey++ are longer (6 digits instead of 4) because users must enter their PIN and the two random digits from NomadiKey++. Logically, longer PINs lead to higher authentication delays. Second, in NomadiKey++ the user often waits for all keys to be highlighted before he can authenticate. This process alone takes 4500 ms, with each key being highlighted for 400 ms. For the sake of comparison, we also show in Figure 5 the authentication delay of NomadiKey and NomadiKey++ without considering the highlighting period.

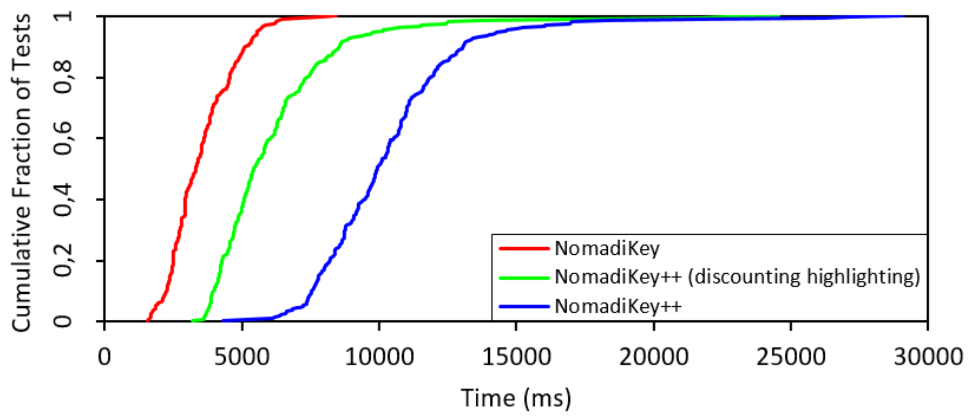

Figure 5. Usability of NomadiKey++.

Finally, we argue that the combination of NomadiKey and NomadiKey++ strikes a good tradeoff between security and usability. Even using smaller keys, the usability of NomadiKey remained at least as good as the usability of random keyboards. Being on average only 1.3s slower than traditional keyboards and with a higher security coefficient. While, despite presenting lower usability, NomadiKey++ presents a considerably higher security level, being the only mechanism with non-trivial security coefficient under shoulder-surfing attacks.

5. Conclusion

In this work, we extend NomadiKey, a mechanism for user authentication on smart devices. NomadiKey places keys in random absolute positions to improve security while keeping the relative positions of keys to preserve usability. In order to increase NomadiKey's security even further, we proposed a security extension called NomadiKey++, that employs out-of-band channels during authentication to thwart shoulder-surfing adversaries. Our analytical and empirical evaluation of NomadiKey and NomadiKey++ indicate they present a good trade-off between security and usability, with NomadiKey++ presenting lower usability for the sake of much stronger security.

References

Arif, A. S. and Mazalek, A. (2013). A Tap and Gesture Hybrid Method for Authenticating Smartphone Users. In *MobileHCI*.

Aviv, A. J., Gibson, K., Mossop, E., Blaze, M., and Smith, J. M. (2010). Smudge Attacks on Smartphone Touch Screens. In *WOOT*.

Bojinov, H. and Boneh, D. (2011). Mobile Token-based Authentication on a Budget. In *HotMobile*.

Cotta, L., Fernandes, A. L., Melo, L. T. C., Saggioro, L. F. Z., Martins, F., Neto, A. L. M., Loureiro, A. A. F., Ítalo Cunha, and Oliveira, L. B. (2016). NomadiKey: User Authentication for Smart Devices based on Nomadic Keys. In *ICC*.

Dell'Amico, M., Michiardi, P., and Roudier, Y. (2010). Password Strength: An Empirical Analysis. In *INFOCOM*.

Egelman, S., Jain, S., Portnoff, R. S., Liao, K., Consolvo, S., and Wagner, D. (2014). Are You Ready to Lock? In *CCS*.

Haque, S. M. T., Wright, M., and Scielzo, S. (2013). Passwords and Interfaces: Towards Creating Stronger Passwords by Using Mobile Phone Handsets. In *SPSM*.

Harbach, M., von Zezschwitz, E., Fichtner, A., Luca, A. D., and Smith, M. (2014). It's a hard lock life: A field study of smartphone (un)locking behavior and risk perception. In *SOUPS*.

Mock, K., Hoanca, B., Weaver, J., and Milton, M. (2012). Real-time Continuous Iris Recognition for Authentication Using an Eye Tracker. In *CCS*.

Neto, A. L. M., Fernandes, A. L., Martins, F., Melo, L. T., Cotta, L., Saggioro, L. F. Z., Loureiro, A. A., and Oliveira, L. B. (2015). Teclanômade: Uma Solução de Autenticação para Usuários de Dispositivos Inteligentes baseada em Teclados Nômades. In *SBSeg*.

Pan, S., Chen, A., and Zhang, P. (2013). Securitas: User Identification Through RGB-NIR Camera Pair on Mobile Devices. In *SPSM*.

Rostami, M., Juels, A., and Koushanfar, F. (2013). Heart-to-Heart (H2H): Authentication for Implanted Medical Devices. In *CCS*.

Wang, H., Lymberopoulos, D., and Liu, J. (2015). Sensor-based User Authentication. In *EWSN*.

Yue, Q., Ling, Z., Fu, X., Liu, B., Ren, K., and Zhao, W. (2014). Blind Recognition of Touched Keys on Mobile Devices. In *CCS*.