NomadiKey: User Authentication for Smart Devices based on Nomadic Keys

Leonardo Cotta, Artur Luis Fernandes, Leandro T. C. Melo, Luiz Felipe Z. Saggioro, Frederico Martins
Antonio L. Maia Neto, Antonio A.F. Loureiro, Ítalo Cunha, Leonardo B. Oliveira

Universidade Federal de Minas Gerais (UFMG) – Belo Horizonte, MG, Brazil

{leonardo.cotta, arturluis, ltcmelo, luizfzsaggioro, fredmbs, lemosmaia, loureiro, cunha, leob}@dcc.ufmg.br

***Abstract*—The growing importance of smart devices calls for effective user authentication mechanisms. We argue that state-of-the-art authentication mechanisms are either vulnerable to known attacks or do not meet usability needs. To address this problem we designed 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 NomadiKey is activated. Besides, NomadiKey preserves usability by maintaining the traditional relative position of keys. We compare NomadiKey with other user authentication mechanisms under different attacks using statistical models and simulation. We also evaluate NomadiKey's usability with 18 users. Our results show that NomadiKey increases security compared to widely-deployed PIN authentication with limited impact on authentication times.**

I. Introduction

Internet of Things (IoT) [1], [2] and its smart devices are ever more present on our day-to-day routine. Current trends indicate these devices will become more widespread, leading to the establishment of smart environments[1].

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 from gigabytes to terabytes of personal, private, and potentially sensitive, user data.

In most cases, a user authenticates onto a smart device inputting a secret previously configured on the device, e.g., typing a PIN or drawing a pattern on the device's unlock screen [3], [4]. Whenever an adversary obtains the authentication secret, he can impersonate the legitimate user [5].

In this paper, we consider two classes of attacks against secret-based authentication (Sec. II), namely *smudge* and *vision* attacks. *Smudge attacks* exploit an image of the screen of a device to reverse engineer the authentication secret [6]. Smudge attacks use digital image processing to identify where the screen was touched from residues on the screen (e.g., oil or dust). Touch positions allows an adversary to infer the numbers in a classic PIN or the pattern used to unlock the phone. *Vision attacks* exploit videos of users during authentication, e.g., videos captured by phone or distant security cameras, to reverse engineer the secret [7]. Vision attacks use computer vision techniques to estimate the position of interface elements (e.g., keys) and the position of the fingers (and shadows) as they approach the screen to infer when and where the screen was touched. Evaluation of smudge and vision attacks show success rates as high as 92% and 91%, respectively.

To address this problem, we present a novel authentication mechanism robust to the aforementioned attacks while incurring negligible increase on authentication speed (Sec. III). NomadiKey, as it is called, places keys at different *absolute* positions on the screen each time, preventing the attacks from inferring which keys are pressed during authentication and improving security. Besides, NomadiKey preserves usability by maintaining *relative* key positions, helping users navigate the keyboard and locate keys.

We compare NomadiKey with four other authentication schemes in face of smudge and vision attacks (Sec. IV). We first present a worst-case model of each attack, then use statistical modeling to quantify security of each authentication mechanism under no attack, under smudge attacks, and under vision attacks. We quantify security as the number of possible authentication secrets that an adversary would need to try in each scenario.

We also compared the usability of our NomadiKey prototype to the usability of classic PIN authentication and of PIN authentication on random keyboards (Sec. V). We had 18 users try all three authentication schemes and measured the amount of time users take to input PINs and the number of errors. Our results show that users can authenticate on NomadiKey almost as fast as in classic PIN authentication and faster than PIN authentication on random keyboards. We also show that NomadiKey increases the security level by several times compared to classic PIN authentication.

We make the following contributions:

1) Present worst-case scenario models of existing attacks on user authentication mechanisms;
2) The conception of NomadiKey, a new authentication mechanism on smart devices;
3) Analytical and empirical analysis of security and usability of NomadiKey and a comparison with other known authentication mechanisms.

We believe NomadiKey strikes a better trade-off between security and usability than other authentication mechanisms in widespread use today. More generally, nomadic keys might

[1] www.forbes.com/sites/louiscolumbus/2013/09/12/idc-87-of-connected-devices-by-2017-will-be-tablets-and-smartphones/

be a useful idea to apply to other existing authentication mechanisms.

II. Background

In this section we discuss recent research on user authentication mechanisms for smart devices (Sec. II-A) and attacks developed to circumvent these mechanisms (Sec. II-B).

A. User authentication mechanisms

User to device authentication 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 [8]. However, an adversary could still simultaneously steal the smart device and the authenticator.

Users can authenticate through their own biological features like fingerprints, voice, or face [9]–[14]. Some previous works propose and evaluate biometric authentication factors from unique body characteristics, like hand characteristics [13], iris recognition [12], ECG measurements [14]. Other previous works consider behavioral patterns to authenticate the user, often transparently [9], [10]; current proposals exploit usage behavior [10] and gesture movement properties [9] to identify users. These mechanisms usually require simple actions from users during authentication and are vulnerable only to few attacks, which makes them an interesting option for smart devices. Biometric schemes, however, require user's biometric features to be stored as secrets in the device. This raises privacy concerns as stealing this secret means stealing private user information, such as the user's fingerprint. Besides, attacks have been shown to be successful on biometric schemes by tricking the authenticator using, for instance, photos, videos or fake fingerprint models [2].

Users can also authenticate themselves by inputting some secret previously configured on the device [5], [15]–[19]. This type of authentication is the most common on smart devices [20]. The security and usability of these mechanisms are heavily affected by the secret chosen; complex secrets increase security but impair usability [21]. Previous work, similar to ours, try to achieve a better trade-off between security and usability by, for instance, motivating users to create longer but easy-to-input passwords [17], using short strokes instead of simply touching buttons [15], considering rhythm of button presses [16], or designing extensions to resist specific attacks [7].

B. Attacks against authentication mechanisms

We consider authentication mechanisms based on what the user knows, i.e., mechanisms where the user inputs a secret touching the device's screen. We model attacks against these authentication mechanisms in the following three classes of increasing sophistication.

Smudge attacks use digital image processing on a photograph of the phone's screen to identify touch locations [6], [22], [23]. Smudge attacks have been evaluated against PIN [23] and pattern [6], [22] authentication, with success rates as high as 70% and 92%, respectively. In practice, smudge attacks may be unable to identify where the screen was touched if the screen is (partially) wiped, e.g., during normal phone use after authentication. Our worst-case scenario considers that smudge attacks can reliably and accurately identify touch locations.

Table I
Worst-case scenario of information available to adversaries

Attack	Touch information		
	Location	Order	Content
Smudge	•		
Vision	•	•	
Shoulder-surfing	•	•	•

Vision attacks exploit videos of users during authentication to reverse-engineer the authentication secret [7]. Videos need not capture the screen contents, just the screen and the user's hand [7], [24]. Videos also need not be high quality, as previous work has shown successful attacks using recordings of a reflection of the device's screen [25]. Vision attacks have been shown to achieve success rates as high as 94% and 98% for direct recordings [7], [24] and 78% for reflections or low quality recordings [25]. Our worst-case scenario considers that vision attacks can reliably and accurately identify both touch locations and touch order.

Shoulder-surfing attacks exploit images captured from a privileged position to obtain screen contents along with user hand movements [26], [27]. While the conventional definition of this attack assumes the adversary is physically standing behind the user and observing the screen, variations include adversaries analyzing recordings made from hidden or public cameras [26], with success rates higher than 95%. Our worst-case scenario considers that shoulder-surfing attacks can identify touch positions, touch order, and screen contents. Our models are summarized in Tab. I.

Finally, smart devices are equipped with various sensors that allow the design of alternative authentication mechanisms (e.g., using rhythm) [28]. Unfortunately, sensors can also be used to perform attacks on authentication mechanisms, for instance, by using the motion sensor, ambient light sensor, camera, and microphone to detect when the user touches the screen and to infer the touch position from properties like device orientation and movement [29], [30]. In this paper we do not consider sensor-based authentication mechanisms and attacks, directly. We note, however, that sensor-based attacks are likely to fit in the smudge or vision categories (Tab. I). We next present NomadiKey, our novel authentication mechanism, and show in Sec. IV that it provides improved security against smudge and vision attacks.

III. NomadiKey

PIN and pattern-based authentication have high usability and are the most common authentication mechanisms used on smart devices [20]. Unfortunately, they are vulnerable to the

[2] blog.lookout.com/blog/2014/09/23/iphone-6-touchid-hack

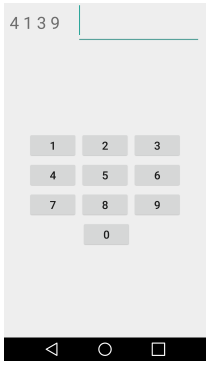

Figure 1. Classic keyboard.

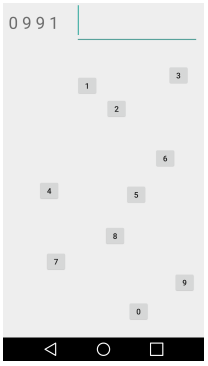

Figure 2. NomadiKey with portrait keyboard's key size.

attacks described in Sec. II-B. An alternative to classic PIN authentication is PEK [7], which randomizes the number on each key on each authentication. PEK increases security level, but it also degrades usability as users are no longer able to build a visual map of their PIN on the keyboard, taking twice as much time to unlock the screen [31]. It has been observed that a typical user will not adopt more secure authentication mechanisms if they introduce complexity [15].

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. Preserving a user's PIN visual map is indispensable.

More precisely, NomadiKey places keys at random *absolute* positions while constraining the *relative* position of keys. Fig. 2 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. As we will show in Sec. V, random *absolute* positions result in a higher security level against smudge and vision attacks, while classic *relative* key positions allows users to authenticate almost as quickly as on a normal keyboard.

A. NomadiKey: Key Position Algorithm

NomadiKey partitions the screen in a grid with as many rows and columns as the authentication keyboard (4 rows and 3 columns for a traditional numeric keyboard). Columns and rows are placed at random locations, and keys are placed in a random location in their corresponding grid cells. We show pseudocode for NomadiKey in Algorithm 1.

Let S_w and S_h be the screen width and screen height, respectively; and let K_w and K_h be the key width and key 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 $(0, 0)$.

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

$$x_{\min}(c) = x(c-1) + \alpha K_w, \tag{1}$$

The αK_w term in Eq. (1), where $\alpha > 1$ is a constant, ensures that there is space for placing keys in column $c-1$. 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 αK_w for each unplaced column to the left of column c. We define $x(0) = -\alpha K_w$. Similarly, the maximum possible starting coordinate $x_{\max}(c)$ for column c is

$$x_{\max}(c) = x(c+1) - \alpha K_w, \tag{2}$$

where we define $x(C+1) = S_w + \alpha K_w$.

NomadiKey then chooses coordinate $x(c)$ where column c starts uniformly distributed between $x_{\min}(c)$ and $x_{\max}(c)$:

$$x(c) = U(x_{\min}(c), x_{\max}(c)). \tag{3}$$

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 above ensures each grid cell is at least αK_w units wide and at least αK_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.

NomadiKey has decreasing flexibility (more constraints) to place any remaining columns as more columns are placed. Left unchecked, this decreasing flexibility could bias columns placed last to specific regions in the screen. This same behavior

Procedure 1 Key Placement

procedure COMPUTE $x_{\min}(c)$
  if column $c-1$ has been placed **then**
    return $x(c-1) + \alpha B_w$
  else
    return $x_{\min}(c-1) + \alpha B_w$
procedure COMPUTE $y_{\max}(r)$
  if row $r+1$ has been placed **then**
    return $y(r+1) - \alpha B_w$
  else
    return $y_{\max}(r+1) - \alpha B_w$
  {*Procedures for* $x_{\max}$ *and* $y_{\min}$ *are analogous*}
procedure NOMADIKEY
  for $c \in$ random$(1, C)$ **do**
    $x(c) \leftarrow U(x_{\min}(c), x_{\max}(c))$
  for $r \in$ random$(1, R)$ **do**
    $y(r) \leftarrow U(y_{\min}(r), y_{\max}(r))$
  for $c \in [1, C]$ **do**
    for $r \in [1, R]$ **do**
      place key in column c, row r at position $U(x(c), x(c+1)), \quad U(y(r), y(r+1))$

Table II
SECURITY COEFFICIENT UNDER DIFFERENT ATTACK MODELS.

MECHANISM	SECURITY COEFFICIENT		
	Safe Operation	Smudge attack	Vision attack
Classic PIN	10^n	$\frac{n!}{\prod_{i=1}^{d} r_i!}$	1
NomadiKey	10^n	$P\frac{n!}{\prod_{i=1}^{d} r_i!}$	P
Random Keyboards	10^n	$\binom{10}{d}\frac{n!}{\prod_{i=1}^{d} r_i!}$	$\frac{10!}{(10-d)!}$
Pattern	$\leq \frac{9!}{(9-n)!}$	2	1
Knock Code	4^n	$\frac{n!}{\prod_{i=1}^{d} r_i!}$	1

n = secret length d = distinct keys r_i = repetitions of key i

applies to rows. To avoid this bias, NomadiKey places columns and rows in random order on each execution.

IV. ANALYTICAL EVALUATION

In this section we compare NomadiKey's security level against classic PIN and pattern authentication, PIN authentication on random keyboards, and LG's new Knock Code using analytical modeling.

We compute each authentication mechanism's *security coefficient*, its number of possible distinct authentication secrets. The inverse of the security coefficient gives the probability of authenticating successfully by entering a random secret.

We compute each authentication mechanism's security coefficients under smudge, vision, and shoulder-surfing attacks, as defined in our attack models in Sec. II. Attacks provide information about the user's secret; they allow an adversary to prune the set of possible secrets and decrease each mechanism's security coefficient.

Tab. II shows closed formulas for the security coefficient of the evaluated mechanisms under safe operation ("no attack") and under different attack models. We express security level as a function of the secret *length*, denoted n. Secret length is the number of key presses for PINs and NomadiKey, the number of knocks for Knock Code, and the number of connected dots for pattern authentication.

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 2×2 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.

Security under smudge attacks. Classic PIN, pattern, and Knock Code authentication have a predictable layout and are vulnerable to smudge attacks. Smudge attacks allow adversaries to identify the location where the screen was touched, but not the order. 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. Let r_i denote the number of times key i was pressed (i.e., number of repetitions) and d denote the number of distinct keys. With no repetitions, the adversary needs to try all $n!$ permutations of the numbers. With repetitions, we divide by the number of permutations of each repeated key, as they are equivalent. Again, Knock Code's security is equivalent to that of a 2×2 keyboard.

For PIN authentication on random keyboards, knowing where the user touched the screen does not provide any information on which keys were pressed. 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. Since knowing the touch position does not provide any information about the pressed key on random keyboards, changing the absolute position of keys does not increase security.

NomadiKey is a middle-ground between PIN authentication on classic and random keyboards. The adversary has to try a number P of key combinations and all possible orderings for each key combination. The number of key combinations in NomadiKey, P, is less than $\binom{10}{d}$ in random keyboards because the adversary can ignore key combinations that contradict NomadiKey's restriction on keeping the relative position of keys. If the smudge reveals two perfectly aligned touches in a vertical line, the adversary can infer that the pressed keys belong to one column and prune the set of possible values for each key. Any PIN that does not fit in the pruned set can ruled out as incompatible with the pressed keys. The value of P depends on the inputs to Algorithm 1; in particular, P depends heavily on key width and key height, as they determine intervals for column and row start positions (Eqs. (1) and (2)). The value of P also depends on secret length, as longer secrets allow for more key combinations but also give away key placement.

For pattern-based authentication under smudge attacks, we note each dot can only be visited once, i.e., the pattern is a Hamiltonian path. An adversary can authenticate by trying the pattern in forward and reverse directions. We also note that although drawing the pattern may lead to intersections (e.g., drawing the two diagonals intersect at the center), each dot is visited the first time it is touched.

Security under vision attacks. Our model for vision attacks gives additional information compared to smudge attacks, so all authentication mechanisms have reduced security coefficients. Vision attacks allow an adversary to know where the screen was touched, and in which order. Classic PIN authentication, pattern-based authentication, and Knock Code have static "keys", so vision attacks give full information over which keys were pressed and in which order.

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.

Security under shoulder-surfing attacks. Our model for shoulder-surfing attacks allows an adversary identify the exact keys touched in NomadiKey and random keyboards, completely reverse-engineering the authentication secret. We note, however, that nomadic keys can be combined with other extensions to PIN authentication to provide increased security level overall as well as robustness against shoulder-surfing attacks. For example, NomadiKey can be extended with strokes, where a user may stroke a key in four directions (up-down or left-right) instead of simply touching it [15]. Combining extensions trades usability for security.

V. Empirical evaluation

A. NomadiKey security

We evaluate the number P of possible key combinations given a set of touch positions empirically. 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 3 to 7. We then run Algorithm 1 one hundred times for each secret to generate different keyboard layouts. We configure NomadiKey with $\alpha = 2$. Finally, we simulate a computer vision attack on each of the 10^7 (secret, layout) pairs to estimate the distribution of P.

Fig. 3 shows the distribution of the value of P over all generated (secret, layout) pairs. We show one curve for secrets with different numbers of distinct keys varying from 3 to 7. Fig. 3 shows P for large keys that are the same size as keys in numeric keyboards used for classic PIN authentication (Fig. 1). We note that PIN-sized keys do not allow any overlap in column start positions, so an attacker can automatically infer the column of the key from the touch position, a conservative configuration scenario for NomadiKey. Even for fixed columns, NomadiKey can still yield a remarkable higher security level than classic keyboards.

Fig. 4 shows similar results for small keys that are the same size as keys in Android's portrait mode typing keyboard (Fig. 2). We observe that, as key size decreases and NomadiKey has more flexibility, security increases significantly. For a typical PIN with four distinct keys, NomadiKey increases the security coefficient by more than 50 times in the median case. Vertical trends for $d = 1$ and $d = 2$ happen for key touch positions where an adversary for which an adversary would need to try all possible d-key combinations, as for PIN authentication on random keyboards.

B. Usability evaluation

To evaluate NomadiKey's usability, we compare it to PIN authentication on classic and random keyboards. We measure usability as *authentication delay*, how long it takes the user to input the secret. We implemented all three authentication mechanisms in an Android application. We adopted a secret

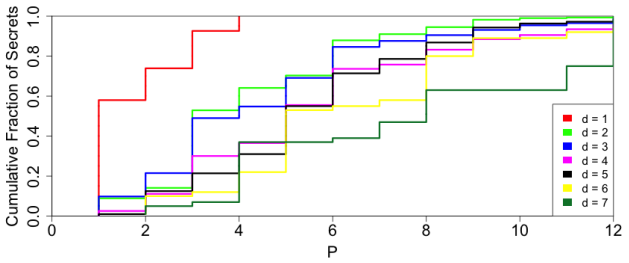

Figure 3. Empirical evaluation of the number P of possible key combinations in NomadiKey for a given set of touch positions and large (PIN-size) keys.

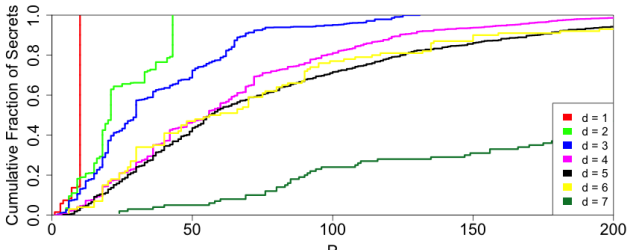

Figure 4. Empirical evaluation of the number P of possible key combinations in NomadiKey for a given set of touch positions and small (qwerty-size) keys.

size of four digits ($n = 4$), based on the average secret length of 4.5 digits found in previous work [32].

Each usability experiment evaluates all three authentication mechanisms in random order. For each authentication mechanism, we generate five different random secrets. For each secret, we ask the user to authenticate with that secret five times (each experiment totals 45 authentications). For each authentication, the user has to press a button before he can see the keyboard. The application stores the time from pressing this button until the user enters the secret or makes an error. Experiments where the user makes a mistake are ignored. Each experiment ends with a quick survey about the usability and perceived security of each keyboard type.

We used two 5" devices to perform usability experiments: an LG G3 and an LG G4. Before starting a usability experiment with a user, we gave a brief overview of the experiment and a brief explanation of NomadiKey. Users did not have any previous experience with NomadiKey. We performed usability experiments with 18 volunteers, 8 female and 10 male. User age varied from 18 to 65 years.

Fig. 5 shows authentication delay for NomadiKey on large keys. We observe users authenticate almost as fast on NomadiKey as on a traditional keyboard, with approximately 6% increase in delay in the median case. Similar to authentication on a random keyboard, minimum authentication times for NomadiKey are not as fast as for traditional keyboards (authentication delays for NomadiKey are at least 1500 ms). We conjecture this is because of the nomadic nature of the keys. By spreading the keys throughout the entire screen, NomadiKey makes it harder for users to (i) locate the first key they need to type and (ii) reach each key. In the median case, users can authenticate on NomadiKey 40% faster than on random keyboards.

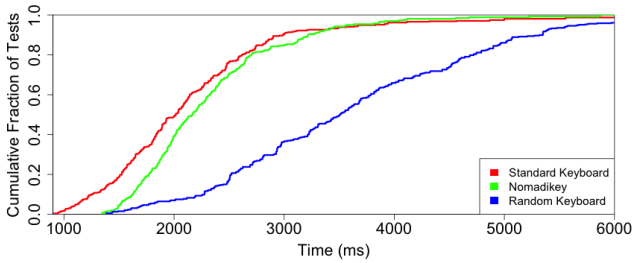

Figure 5. Usability of NomadiKey and PIN authentication on large keys.

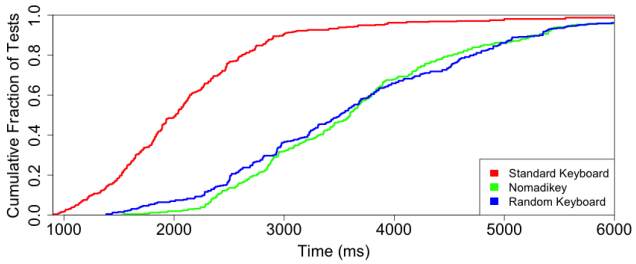

Figure 6. Usability of NomadiKey and PIN authentication on small keys.

Fig. 6 shows similar results for NomadiKey using small keys. We observe that decreasing key size also result in longer authentication delays. Besides, even for small keys, users can still authenticate as fast on NomadiKey as on random keyboards.

Finally, we note several (62%) users reported they were willing to try NomadiKey in the survey. Several (75%) users reported NomadiKey to be more usable than random. Some (44%) users also had the (incorrect) impression and reported they thought NomadiKey more secure than random keyboards.

VI. Conclusion

Smart devices are ubiquitous and the amount of personal, often sensitive, information they carry impose a need for higher security. Unfortunately, users are reluctant to use authentication mechanisms that impact usability. In this work, we have proposed NomadiKey, a new 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. Our analytical and empirical evaluation of NomadiKey indicate it is a good trade-off between PIN authentication on traditional and random keyboards. As future work, we plan to study how to choose key sizes to maximize security and usability.

Acknowledgements

We would like to thank Carlos Toneto, Andre Graziani, Pablo Marcondes, Vitor Paisante, Ivan Nunes, Ronaldo Resende and Lucas Grossi for fruitful discussions and feedback. This work was partially funded by CNPq, CAPES, and FAPEMIG.

References

[1] K. Ashton, "That 'Internet of Things' Thing," *RFiD Journal*, vol. 22, no. 7, pp. 97–114, 2009.

[2] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[3] R. J. Anderson, *Security Engineering: A Guide to Building Dependable Distributed Systems*. Wiley Publishing, 2008.

[4] D. Todorov, *Mechanics of User Identification and Authentication: Fundamentals of Identity Management*. CRC Press, 2007.

[5] S. Wiedenbeck, J. Waters, L. Sobrado, and J.-C. Birget, "Design and Evaluation of a Shoulder-Surfing Resistant Graphical Password Scheme," in *AVI*, 2006.

[6] A. J. Aviv, K. Gibson, E. Mossop, M. Blaze, and J. M. Smith, "Smudge Attacks on Smartphone Touch Screens," in *WOOT*, 2010.

[7] Q. Yue, Z. Ling, X. Fu, B. Liu, K. Ren, and W. Zhao, "Blind Recognition of Touched Keys on Mobile Devices," in *CCS*, 2014.

[8] H. Bojinov and D. Boneh, "Mobile Token-based Authentication on a Budget," in *HotMobile*, 2011.

[9] A. De Luca, A. Hang, F. Brudy, C. Lindner, and H. Hussmann, "Touch Me Once and I Know It's You!: Implicit Authentication Based on Touch Screen Patterns," in *CHI*, 2012.

[10] M. Jakobsson, E. Shi, P. Golle, and R. Chow, "Implicit Authentication for Mobile Devices," in *HotSec*, 2009.

[11] J. Liu, Z. Wang, L. Zhong, J. Wickramasuriya, and V. Vasudevan, "uWave: Accelerometer-based Personalized Gesture Recognition and Its Applications," in *PerCom*, 2009.

[12] K. Mock, B. Hoanca, J. Weaver, and M. Milton, "Real-time Continuous Iris Recognition for Authentication Using an Eye Tracker," in *CCS*, 2012.

[13] S. Pan, A. Chen, and P. Zhang, "Securitas: User Identification Through RGB-NIR Camera Pair on Mobile Devices," in *SPSM*, 2013.

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

[15] A. S. Arif and A. Mazalek, "A Tap and Gesture Hybrid Method for Authenticating Smartphone Users," in *MobileHCI*, 2013.

[16] Y. Chen, J. Sun, R. Zhang, and Y. Zhang, "Your Song Your Way: Rhythm-based Two-factor Authentication for Multi-touch Mobile Devices," in *INFOCOM*, 2015.

[17] S. M. T. Haque, M. Wright, and S. Scielzo, "Passwords and Interfaces: Towards Creating Stronger Passwords by Using Mobile Phone Handsets," in *SPSM*, 2013.

[18] I. Jermyn, A. Mayer, F. Monrose, M. K. Reiter, and A. D. Rubin, "The Design and Analysis of Graphical Passwords," in *USENIX Security*, 1999.

[19] V. Roth, K. Richter, and R. Freidinger, "A PIN-entry Method Resilient Against Shoulder Surfing," in *CCS*, 2004.

[20] S. Egelman, S. Jain, R. S. Portnoff, K. Liao, S. Consolvo, and D. Wagner, "Are You Ready to Lock?" in *CCS*, 2014.

[21] M. Dell'Amico, P. Michiardi, and Y. Roudier, "Password Strength: An Empirical Analysis," in *INFOCOM*, 2010.

[22] P. Andriotis, T. Tryfonas, and Z. Yu, "Breaking the Android Pattern Lock Screen with Neural Networks and Smudge Attacks," in *WiSec*, 2014.

[23] Y. Zhang, P. Xia, J. Luo, Z. Ling, B. Liu, and X. Fu, "Fingerprint Attack Against Touch-enabled Devices," in *SPSM*, 2012.

[24] D. Shukla, R. Kumar, A. Serwadda, and V. V. Phoha, "Beware, Your Hands Reveal Your Secrets!" in *CCS*, 2014.

[25] R. Raguram, A. M. White, D. Goswami, F. Monrose, and J.-M. Frahm, "iSpy: Automatic Reconstruction of Typed Input from Compromising Reflections," in *CCS*, 2011, pp. 527–536.

[26] F. Maggi, A. Volpatto, S. Gasparini, G. Boracchi, and S. Zanero, "Poster: Fast, Automatic iPhone Shoulder Surfing," in *CCS*, 2011.

[27] F. Schaub, R. Deyhle, and M. Weber, "Password Entry Usability and Shoulder Surfing Susceptibility on Different Smartphone Platforms," in *MUM*, 2012.

[28] I. T. Fischer, C. Kuo, L. Huang, and M. Frank, "Short Paper: Smartphones: Not Smart Enough?" in *SPSM*, 2012.

[29] L. Simon and R. Anderson, "PIN Skimmer: Inferring PINs Through the Camera and Microphone," in *SPSM*, 2013.

[30] R. Spreitzer, "PIN Skimming: Exploiting the Ambient-Light Sensor in Mobile Devices," in *SPSM*, 2014.

[31] Q. Yue, Z. Ling, X. Fu, B. Liu, K. Ren, and W. Zhao, "Blind Recognition of Touched Keys: Attack and Countermeasures," *CoRR*, vol. abs/1403.4829, 2014.

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