Received: 1 March 2017 | Revised: 17 July 2017 | Accepted: 12 August 2017

DOI: 10.1002/nem.1998

SPECIAL ISSUE PAPER

NomadiKey: User authentication for smart devices based on nomadic keys

Artur Souza | Ítalo Cunha | Leonardo B Oliveira

Department of Computer Science, Universidade Federal de Minas Gerais, Belo Horizonte, Brazil

Correspondence
Artur Souza, Department of Computer Science, Universidade Federal de Minas Gerais, Belo Horizonte, MG, 31270-901, Brazil.
Email: arturluis@dcc.ufmg.br

Funding information
CAPES; CNPQ; FAPEMIG

Summary

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 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 that uses out-of-band channels to thwart shoulder-surfing adversaries. We compare NomadiKey with other user authentication mechanisms under different attacks using statistical models and simulation. We also evaluate NomadiKey's usability with 20 users. Our results show that NomadiKey increases security compared to widely deployed PIN authentication with limited impact on authentication times.

1 | INTRODUCTION

The Internet of Things (IoT)[1] denotes a paradigm where users are surrounded by computing elements and sensors attached to all sorts of everyday objects. These elements, called "things," enable a series of new and ubiquitous applications that impact every aspect of users' day-to-day routine.[2]

With the ongoing expansion of IoT deployment and functionality, it is paramount to ensure devices are properly secured; otherwise, adversaries will easily exploit the devices or the applications they provide.[3] The recent Mirai botnet attacks,[*] which disrupted a large fraction of the Internet for several hours, are a clear example that IoT devices are already considered valuable targets and that failing to ensure their security leads to severe consequences.

Notably, ensuring the security of smart IoT devices, like smartphones and tablets, is critical. These devices are vastly more powerful than other IoT devices (like wearables or sensors) and they often serve as gateways for users to control and manage their other IoT devices.[4,5] These 2 aspects, together with the massive (and bound to grow) amount of private information and sensitive applications currently available on these devices turn security into a paramount concern.[3] In particular, stronger user-to-device authentication mechanisms are paramount to prevent unauthorized access to personal smart devices and, consequently, to other IoT devices.[6]

Usability is almost, if not equally, as important as security in authentication mechanisms. It has been shown that users will not use stronger authentication mechanisms if they are slow or hard to use.[7] However, to achieve higher

[*]For example, see https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet.

security, authentication mechanisms often sacrifice usability.[8] This implies on a trade-off between security and usability that hinders the design of new secure authentication mechanisms.

The design is further handicapped by IoT's stronger adversary model. Adversaries may take advantage of the smart device's functionalities (eg, their sensors or capacitive screens) to launch attacks that were not previously possible on authentication mechanisms.[9] The inherent mobility of these devices also becomes a drawback, as users may lead these devices into unsafe areas where it is easier for adversaries to launch attacks or steal the device.[9] If the adversary obtains the authentication secret and steals the device, he can impersonate the user and access the device.[10]

In this paper, we consider 3 classes of attacks against authentication mechanisms (Section 2), namely, *smudge*, *computer vision*, and *shoulder-surfing* attacks. Smudge attacks exploit residues left on the screen after authentication to identify where the screen was touched and infer the numbers in a classic personal identification number (PIN) or the pattern used to unlock the phone. 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. Shoulder-surfing adversaries observe the user authenticating, or videos of the user authenticating, to obtain everything the user input during authentication. We consider these attacks because of their simplicity and effectiveness on existing authentication mechanisms. Evaluation of these attacks shows success rates as high as 92%,[11] 98%,[12] and 95%,[13] respectively.

To defend against the aforementioned attacks, we present a novel authentication mechanism robust to these attacks while incurring a negligible increase in authentication speed (Section 3). NomadiKey,[14] 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. To increase NomadiKey's security against attacks, especially against shoulder-surfing attacks, we also present a security extension for NomadiKey that uses vibrations as an out-of-band channel during authentication. Preventing adversaries from retrieving the entire authentication secret through shoulder-surfing attacks.

We evaluate NomadiKey regarding its security and usability. We compare NomadiKey's security with 5 other authentication schemes in face of smudge, vision, and shoulder-surfing attacks (Section 4), and we compare the usability of NomadiKey with the usability of classic PIN authentication and of PIN authentication on random keyboards. Our results indicate that NomadiKey strikes an interesting trade-off between security and usability. With the usability of NomadiKey being almost as good as that of classic PINs and its security several times higher. We believe the small usability overhead may discourage security-oblivious users from adopting NomadiKey,[7,15] but we claim it is an interesting solution for users that require stronger security and are willing to sacrifice usability for it.

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. the design of a security extension to NomadiKey to protect users against shoulder-surfing attacks;
4. analytical and empirical analysis of security and usability of NomadiKey with and without the security extension and a comparison with other known authentication mechanisms.

The rest of this paper is organized as follows: Section 2 presents a background on user authentication on smart devices. Section 3 describes NomadiKey, our solution. Sections 4 and 5 present, respectively, security and usability evaluations of NomadiKey as well as a comparison with other known authentication mechanisms. Finally, in Section 6, we conclude this work.

2 | BACKGROUND

In this section, we discuss recent research on user authentication mechanisms for smart devices (Section 2.1) and attacks developed to circumvent these mechanisms (Section 2.2).

2.1 | User authentication mechanisms

User to device authentication mechanisms can be divided into 3 categories: (1) something the user has, (2) something the user is, and (3) something the user knows. In this section, we highlight the state of the art of each category.

TABLE 1 Summary of user authentication categories

Category	Authenticator	Security	Usability	Flexibility	Privacy
Physical	Token	Low	High	Low	High
Biometric	Biometric features	Very High	High	Very Low	Low
Knowledge	Secret	Low to high	Medium to low	High	High

Table 1 summarizes the work presented in this section regarding the type of secret used, usability, and security levels, flexibility (how easy it is to create, change, or delete secrets), and privacy of the solution (how much information is disclosed through the secret).

2.1.1 | Something you have

Users can authenticate using a physical authenticator they have, like a token or smart card. In this category, Bojinov and Boneh[16] have proposed two token based authentication mechanisms for smartphones. In the first mechanism, the user carries a token capable of modulating a digital signal through changes in a magnetic field. The user authenticates by triggering the signal using his token. The second mechanism works in a similar fashion but uses sound generated by a buzzer to transmit signals to the smartphone instead.

Limitations. While the authentication process is often simple, there are a few drawbacks to token-based authentication.[17] First, users are required to carry their token at all times, which may become a hassle. Second, if a user forgets or loses his token, he is locked out of his device. Third, if an adversary wishes to impersonate the user, he may simply simultaneously steal the authenticator and the smart device. Finally, it is not easy to replace a lost or stolen authenticator. Few works propose token-based authentication for smart devices, possibly because of the aforementioned drawbacks.

2.1.2 | Something you are

Users can also authenticate through their own biometric features like fingerprint, voice, or behavior.[18-23] Pan et al,[22] for instance, propose using hand characteristics like skin tone, palm length, finger valleys, and finger width to authenticate users. They rely on an RGB-NIR camera pair to retrieve information from users' hands.

Mock et al[21] propose an authentication mechanism based on iris recognition. The authors propose using a commercial eye tracker to continuously scan the user's iris to achieve continuous authentication. That is, the device locks itself at any time if it decides that the current user is not legitimate. As a complement to their work, the authors suggest combining iris recognition with gaze tracking for more precise authentication.

Zhang et al[24] use phonemes to ensure liveness for voice authentication. The authors use smartphones' microphones to obtain users' voice samples and compare it to a stored recording for the sake of authentication. Their proposed system is able to differentiate live users from recordings based on nuances of the human speech producing process. Since live users and recorders generate sounds differently, their system is able to determine when the voice sample comes from a recording and deny authentication.

Jakobsson et al[19] propose analyzing the smartphone usage to authenticate users implicitly. In their scheme, the device creates a usage pattern based on the actions the user often conducts in his routine to, later, authenticate the user based on this routine. They create the usage pattern based on user location, phone activity (when the phone is used) and application activity (which applications are used and when). Their authentication mechanism is continuous and implicit.

Similarly, De Luca et al[18] analyze touch patterns to improve pattern-based authentication. The authors claim that the way in which users draws their patterns is unique and can be used to strengthen pattern-based authentication. To construct the user's touch pattern, they consider the touch size, movement speed, gesture time, pressure applied, and touch coordinates.

Finally, Jie et al[23] propose a sensor-based authentication scheme. They expand on existing gesture-based authentication by using the device's sensors. Their scheme uses touch sensors to identify gesture movements and hand geometry, size, and movement. At the same time, the device's accelerometer and gyroscope are used to record device displacement and rotation during authentication. All the information extracted is combined to form a user fingerprint model, later used during authentication. To optimize authentication, the authors suggest 2 gestures that help sensors extract the maximum amount of information possible.

Limitations. Biometric 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. These mechanisms, 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.[25] Lastly, biometric factors are often unique and final, meaning a user cannot change the secret (his biometric features) once they are leaked and must use the same secret on all systems that rely on the same feature.

2.1.3 | Something you know

Users can also authenticate themselves by inputting some secret previously configured on the device.[7,26-30] This type of authentication is the most common on smart devices.[31] The security and usability of these mechanisms are heavily affected by the secret chosen; complex secrets increase security but impair usability.[32] Previous work, similar to ours, try to achieve a better trade-off between security and usability. To achieve this, some authors propose novel authentication mechanisms, while others propose improvements on existing mechanisms.

Chen et al,[26] for instance, propose a novel rhythm-based authentication mechanism. In their mechanism, users authenticate by performing a series of taps and slides on the device screen, following a rhythm. The mechanism also takes in consideration behavioral metrics during authentication, being a combination of biometric and knowledge-based authentication.

Dunphy and Yan[33] have proposed an improvement on graphical password schemes based on background images. They claim that users often choose weak or predictable graphical passwords that undermine the security of the authentication mechanism. To address this issue, they propose adding background images to discourage predictable drawings.

Uellenbeck et al[34] aim to improve pattern-based authentication by motivating users to create stronger patterns. The authors identify that most users create similar patterns by making default choices, for instance, starting the pattern in the top-left button. They show that this biases the patterns created and enables dictionary attacks. To address this, they propose several different layouts for the distribution of dots on the screen. They aim for layouts that do not compel users to follow predictable patterns, leading to stronger and more diverse patterns.

Similarly, Haque et al[27] try to improve password-based authentication by motivating users to create stronger passwords. They claim that users often avoid adding special characters on smart devices passwords because of the hassle of adding them. To address this, they design a new keyboard layout that facilitates the input of special characters during password creation and entry, to motivate users to include special characters more often.

Krombholz et al[30] propose using pressure sensitive touchscreens to improve PIN-based authentication. The authors propose a PIN-based authentication mechanism that allows users to press keys in 2 different intensities. When creating and entering their PINs, users can press a key strongly or lightly. The goal of their solution is to thwart shoulder-surfing adversaries if adversaries are unable to observe the pressure applied to the touches.

Similarly, Arif and Mazalek[7] propose an improvement to PIN-based authentication based on strokes. They propose an authentication mechanism that allows users to stroke keys in a certain direction, rather than simply tapping the keys. Their solution increases the password character space from 10 to 50 distinct entries.

Yue et al[8] propose an attack on authentication mechanism and, based on the attack, design 2 authentication mechanisms as countermeasures. The attack proposed is the computer vision attack we consider in this work (Section 2.2). To thwart this attack, they propose 2 novel keyboard layouts. Their first mechanism simply randomizes the position of each key in the keyboard, preventing computer vision attacks from recovering the touched key. The second mechanism not only randomizes key positions but also applies a Brownian motion to them, making keys move around the screen in a fixed region. Both attacks succeed in preventing computer vision attacks at the expense of high usability loss.

Limitations. While there is much work on novel or improved user authentication mechanisms, we argue that existing mechanisms still do not fully address the user's needs. In particular, existing knowledge-based mechanisms either fail to protect the user from known attacks (Section 2.2) or fail to provide good usability to users. We conceive NomadiKey and its security extension to address these shortcomings, ie, provide both security and usability at a reasonable level.

2.2 | Attacks against authentication mechanisms

We consider authentication mechanisms based on what the user knows, ie, mechanisms where the user inputs a secret by touching the device's screen. We model attacks against these authentication mechanisms in the following 3 classes of increasing sophistication.

TABLE 2 Worst-case model of information available to adversaries

	Touch information		
Attack	**Location**	**Order**	**Content**
Smudge	●		
Vision	●	●	
Shoulder-surfing	●	●	●

Smudge attacks use digital image processing on a photograph of the phone's screen to identify touch locations.[11,15,35] Smudge attacks have been evaluated against PIN[35] and pattern[11,15] authentication, with success rates as high as 70% and 92%, respectively. Our worst-case model considers that smudge attacks can reliably and accurately identify touch locations after authentication.

Vision attacks exploit videos of users during authentication to reverse-engineer the authentication secret.[36] Videos need not capture the screen contents, just the screen and the user's hand.[12,36] Videos also need not be high quality, as previous work has shown successful attacks using recordings of a reflection of the device's screen.[37] Vision attacks have been shown to achieve success rates as high as 94% and 98% for direct recordings[12,36] and 78% for reflections or low-quality recordings.[37] Our worst-case model 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.[13,38] 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,[13] with success rates higher than 95%. Our worst-case model considers that shoulder-surfing attacks can identify touch positions, touch order, and screen contents. Our models are summarized in Table 2.

Finally, smart devices are equipped with various sensors that allow the design of alternative authentication mechanisms (eg, using rhythm).[9] 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.[39,40] In this paper, we do not consider sensor-based attacks directly. We note, however, that sensor-based attacks are likely to fit in the smudge or vision categories (Table 2).

3 | NOMADIKEY

The PIN and pattern-based authentication have high usability and are the most common authentication mechanisms used on smart devices.[31] Unfortunately, they are vulnerable to the attacks described in Section 2.2. A classic PIN keyboard is shown in Figure 1. An alternative to classic PIN authentication is password encryption key,[36] which randomizes the number on each key for each authentication. Figure 2 shows an example of random keyboard. Privacy enhancing keyboard 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.[8] It has been observed that a typical user will not adopt more secure authentication mechanisms if they introduce complexity.[7]

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. In other words, NomadiKey can be seen as a middle ground between classic PIN authentication and random keyboard PIN authentication.

More precisely, NomadiKey places keys at random *absolute* positions while constraining the *relative* position of keys. Figure 3 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 Sections 4 and 5, random *absolute* positions result in a higher security level against smudge and vision attacks, while classic *relative* key positions allow users to authenticate almost as quickly as on a normal keyboard.

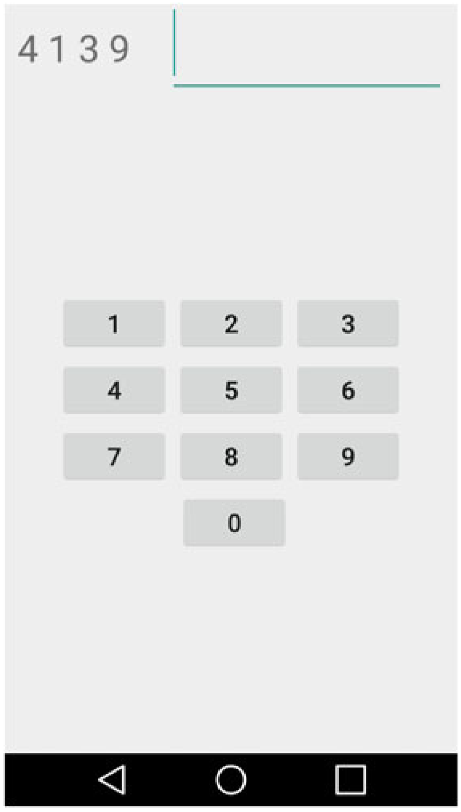

FIGURE 1 Classic keyboard

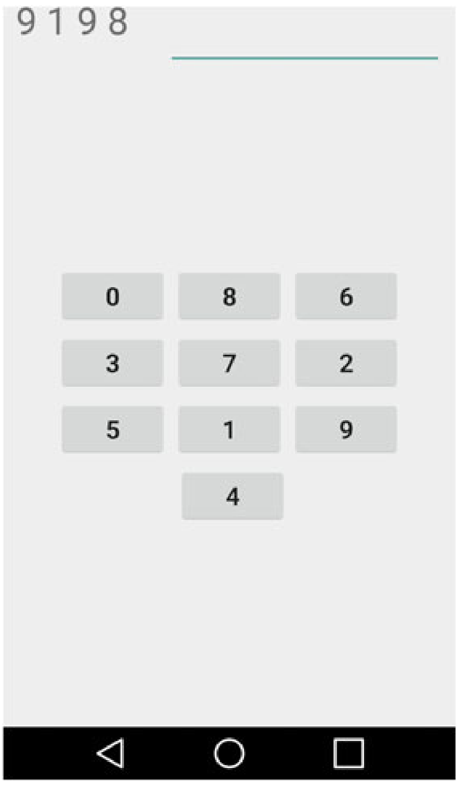

FIGURE 2 Random keyboard

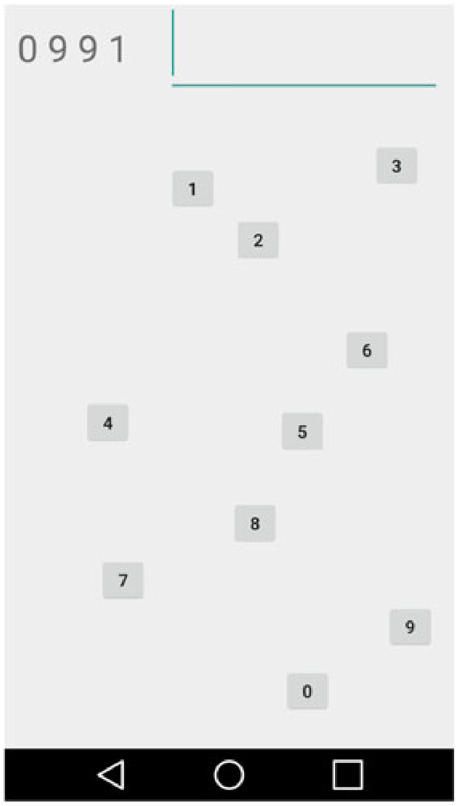

FIGURE 3 NomadiKey

3.1 | 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 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, ie, 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

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

The αB_w term in Equation 1 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

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

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 $x_{\max}(c)$:

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

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 $y_{\max}(r)$.

Algorithm 1 Key placement

```
1:  $C \leftarrow$ number of columns
2:  $R \leftarrow$ number of rows
3:  $B_w \leftarrow$ button width
4:  $B_h \leftarrow$ button height
5:  $U(a, b) \leftarrow$ uniformly distributed random number between a and b
6:  procedure COMPUTE $x_{\text{MIN}}$ $(c)$
7:      if column $c - 1$ has been placed then
8:          return $x(c-1) + \alpha B_w$
9:      else
10:         return $x_{\min}(c-1) + \alpha B_w$
11: procedure COMPUTE $y_{\text{MAX}}$ $(r)$
12:     if row $r + 1$ has been placed then
13:         return $y(r+1) - \alpha B_w$
14:     else
15:         return $y_{\max}(r+1) - \alpha B_w$
    {Procedures for $x_{\max}$ and $y_{\min}$ are analogous}
16: procedure NOMADIKEY
17:     for $c \in \text{random}(1, C)$ do
18:         $x(c) \leftarrow U(x_{\min}(c), x_{\max}(c))$
19:     for $r \in \text{random}(1, R)$ do
20:         $y(r) \leftarrow U(y_{\min}(r), y_{\max}(r))$
21:     for $c \in [1, C]$ do
22:         for $r \in [1, R]$ do
23:             place key in column $c$, row $r$ at position
    $U(x(c), x(c+1)), \quad U(y(r), y(r+1))$
```

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.

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 on the screen. This same behavior applies to rows. To avoid this bias, NomadiKey places columns and rows in random order on each execution.

3.2 | NomadiKey ++: shoulder-surfing protection

By randomizing the keys' absolute position, NomadiKey strikes a higher security level against smudge and vision attacks (Section 4). 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 uses 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 ++ provides increased protection against shoulder-surfing, smudge, and computer vision attacks at the cost of lower usability. Because of this extra usability impact, we conceived it as an extension, rather than a standalone authentication mechanism. That is, it is not meant to be used as the sole authentication mechanism of the device; instead, it is activated when the user wishes for stronger authentication. For instance, when the user thinks someone might be observing him use his device.

NomadiKey ++ works as follows. When it is triggered, the user is presented with a NomadiKey keyboard, as described in Section 3.1. As soon as the keyboard appears, NomadiKey++ starts highlighting keys one by one. In 2 random digits,

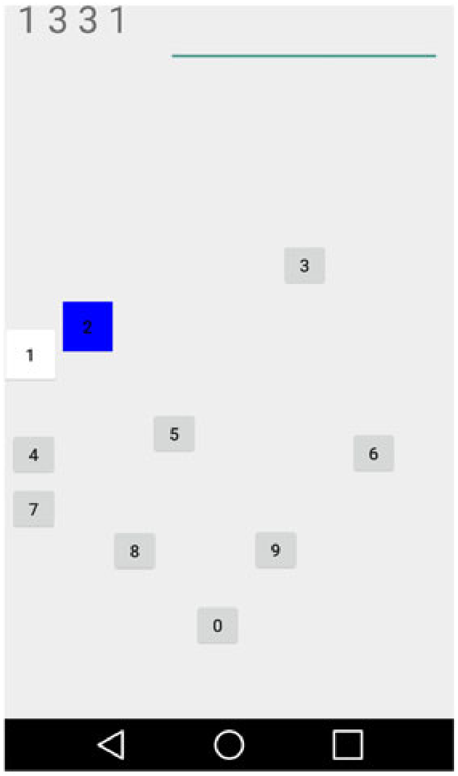

FIGURE 4 NomadiKey ++

the device vibrates while highlighting the key. After all keys have been highlighted, the user authenticates himself by entering his secret and the 2 keys indicated by the vibrations. The extra keys can be included in any order and position, even between digits of the secret. Figure 4 shows an example of NomadiKey++ highlighting the key 2.

For example, suppose the device vibrates on digits 4 and 7 and the user's secret is 1331. To authenticate, the user may enter any string that includes the original secret and the 2 extra digits in any order and position (eg, 133147, 471331, 143371, etc). An adversary attempting a shoulder-surfing attack will see only the entered sequence (eg, 143371) and will not know which digits belong to the user's PIN and which are random digits included by NomadiKey ++.

In the following sections, we evaluate and compare the security and usability of NomadiKey and NomadiKey ++ with other authentication mechanisms. In Section 4, we show an analytical evaluation and in Section 5, an empirical evaluation.

4 | ANALYTICAL EVALUATION

In this section, we compare the security of NomadiKey and NomadiKey ++ with classic and random PIN keyboard layouts, pattern-based authentication (Figure 5), LG's new Knock Code[†] (Figure 6), and Samsung's Digital Door Lock[‡] (Figure 7). Before presenting the comparison, we briefly explain LG's Knock Code and Samsung's Digital Door Lock.

LG's Knock Code allows users to authenticate by performing a series of taps ("knocks") on the device's screen. These taps can be performed in 1 of 4 quadrants (top left, top right, bottom left, and bottom right) and the user may use from 3 to 8 taps. Knock Code allows users to perform the taps anywhere on the screen, as long as the quadrants are clearly defined.

Samsung's Digital Lock uses random digits to increase the security of its PIN-based authentication against smudge attacks. When the user begins its authentication process, Digital Lock lights up 2 of its digits. To authenticate, the user first presses the 2 lit digits and a confirmation button and then enters his PIN. The goal of this feature is to prevent adversaries from retrieving the user's PIN from fingerprint marks left on the door lock, ie, it aims to prevent smudge attacks.

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

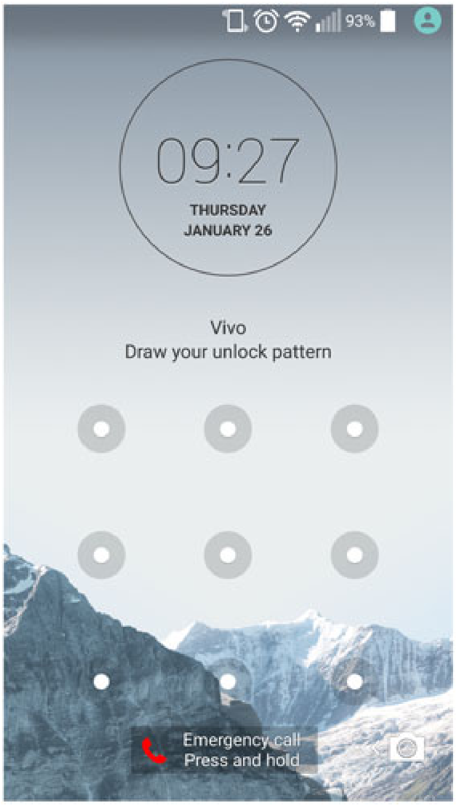

FIGURE 5 Pattern authentication

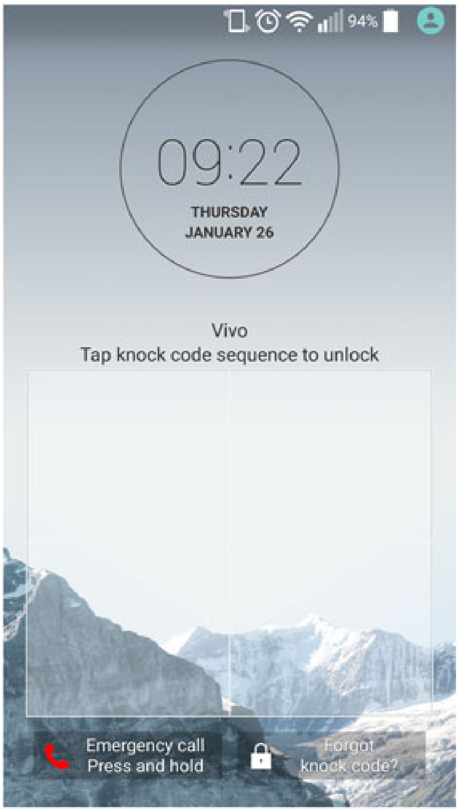

FIGURE 6 Lg's Knock Code

FIGURE 7 Samsung's Digital Lock

TABLE 3 Security coefficient under different attack models

	Security coefficient			
Mechanism	**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				

We compare all these authentication mechanisms through their *security coefficient*, their number of possible distinct authentication secrets. The inverse of the security coefficient gives the probability of authenticating successfully by entering a random secret. 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 2. Each attack provides information about the user's secret; they allow an adversary to prune the set of possible secrets and decrease each mechanism's security coefficient.

Table 3 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 *lengthn*, 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 that we consider a more generic version of Digital Lock that adds f random digits instead of 2.

4.1 | 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 9 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. For instance, a pattern starting at the top-left dot followed by the bottom-right dot, without passing through at least 1 other dot first, is impossible. 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.

4.2 | 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 1 number in the PIN. After identifying the numbers in the PIN, the adversary needs to guess the order in which they should be entered.

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. The only information revealed to the adversary 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. 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 its security.

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 random keyboard's $\binom{10}{d}$ 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 2 perfectly aligned touches in a vertical line, the adversary can infer that the pressed keys belong to 1 column and prune the set of possible values for each key. Any PIN that does not fit in the pruned set can be 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 (Equations 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. In Section 5, we show empirical values of P for different secret lengths.

For pattern-based authentication under smudge attacks, we note that each dot can only be visited once, ie, the pattern is a Hamiltonian path. An adversary can authenticate by inputting the pattern seen in the smudge in forward and reverse directions. We also note that although drawing the pattern may lead to intersections (eg, drawing the 2 diagonals intersect at the center), each dot is visited the first time it is touched and the order of the touches can be easily identified in the smudge trail. An attacker needs only try which direction the dots must be connected.

In Digital Lock and NomadiKey ++, f random digits are entered together with the PIN during authentication. When analyzing the smudge, an adversary is unable to distinguish the digits belonging to the PIN from the random digits added by the authentication mechanism. 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, where some of the digits repeat and others do not. We note that the number of possibilities for removing f digits from the $n + f$ set is the same as the possibilities of choosing the real n PIN digits from the $n + f$ set. The number of possible combinations can be calculated using generating functions.[41]

With generating functions, the combinations can be modeled as a product of equations of the form $\sum_{i=0}^{r_i} x^i$, where r_i is the number of repetitions of the key i. The expanded form of this product is an equation of order $n + f$. The number of combinations of f elements that can be taken from a set of $n + f$ elements is given by the coefficient of the x^f component in the expanded form of the equation.

For instance, consider that the smudge attack reveals the $n + f$ set 135578, with $n = 4$ and $f = 2$. The number of combinations of f taken from $n + f$ can be computed with the equation:

$$(1 + x + x^2) * (1 + x)^4 = x^6 + 5x^5 + 11x^4 + 14x^3 + 11x^2 + 5x + 1. \tag{4}$$

Since the coefficient of $x^{f=2}$ is 11, there are 11 distinct combinations of 2 digits in this set.

In our evaluation, we use G_f^{n+f} to express the number of possibilities of choosing f digits from an $n+f$ subset, considering key repetitions.

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. 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 on 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.

4.3 | 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, 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 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.

4.4 | 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 nontrivial 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 the numbers that vibrated from the numbers that were just highlighted and, thus, cannot distinguish between random and real PIN digits. Because of this, the security coefficient for NomadiKey ++ under shoulder-surfing remains G_f^{n+f}.

Finally, we recall that the nomadic keys of NomadiKey and the out-of-band channel of NomadiKey ++ can be combined with other existing authentication mechanisms to increase their security level and robustness against attacks. For instance, NomadiKey and NomadiKey ++ can be combined with pressure-sensitive touchscreens,[30] doubling the possibilities for each digit of the secret and increasing robustness against all aforementioned attacks. NomadiKey and NomadiKey ++ can also be combined with strokes, where a user may stroke a key in four directions (up, down, left, or right) instead of simply touching it.[7] Strokes increase the possibility for each digit to 50 instead of 10. It is expected, however, that combining these mechanisms will increase NomadiKey's security level at the expense of usability.

5 | 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 parameters P and G_f^{n+f}. We also evaluate and compare the usability of NomadiKey, NomadiKey ++, classic PIN keyboards, and random PIN keyboards using prototypes we developed for each mechanism.

5.1 | Security evaluation

To complete our security evaluation of NomadiKey and NomadiKey ++, we evaluate the values of P and G_f^{n+f} empirically. Recall that P is the number of possible key combinations given a set of touch positions. Also, recall that G_f^{n+f} is the number of possibilities for the adversary to remove the random digits from the user's PIN in NomadiKey ++ and Digital Lock.

To estimate P, we implemented a purpose-specific program that simulates a computer vision attack on NomadiKey. Our simulator receives a set of touch positions and the order of these touches as input (the information obtained through a computer vision attack). Based on this information, it prunes the set of possible user secrets to remove all secrets that are not possible (eg, 5-digit secrets when there were only 4 touches) or that contradict NomadiKey's button placement restrictions. The number of remaining possible secrets represents the mechanism's security coefficient and, in this case, the parameter P.

To evaluate the distribution of 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 100 different keyboard layouts for each secret using Algorithm 1 with $\alpha = 2$, ensuring each row (column) is at least 2 buttons wide (high). Finally, we simulate a computer vision attack on each of the 10^7 (secret, layout) pairs using our simulator.

The size of the buttons used by NomadiKey greatly impact its security and usability. Bigger buttons are easier to find and reach, leading to fewer errors and shorter authentication time, consequently, better usability. Conversely, smaller buttons are easier to distribute throughout the screen, leading to a more diverse set of keyboard layouts, consequently, better security. Because of the impact of button size on security, we evaluate P with small and big buttons.

Figure 8 shows the distribution of P for large keys that are the same size as keys in keyboards used for classic PIN authentication (Figure 1). The PIN-sized keys do not allow any overlap in column start positions and adversaries can automatically infer the column of each key from the touch position, a conservative configuration scenario for NomadiKey. We note, however, that even with fixed columns, NomadiKey can still yield a remarkable higher security level than classic keyboards.

Figure 9 shows the distribution of P for small keys that are the same size as keys in Android's portrait mode typing keyboard (Figure 3). We observe that, as key size decreases and NomadiKey flexibility increases, security increases significantly. For PINs with 4 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 touch positions where an adversary would have to try all possible d-key combinations, as for PIN authentication on random keyboards.

To conclude the evaluation of the security level of NomadiKey ++, we show in Table 4 the average value of G_f^{n+f}. We generate all possible combinations of secret length (n) and extra values (f) and compute the average. As expected, the security level increases together with both n and f, with f increasing the security level more sharply than n. That is, there is a sharper increase on G_f^{n+f} between columns than between rows.

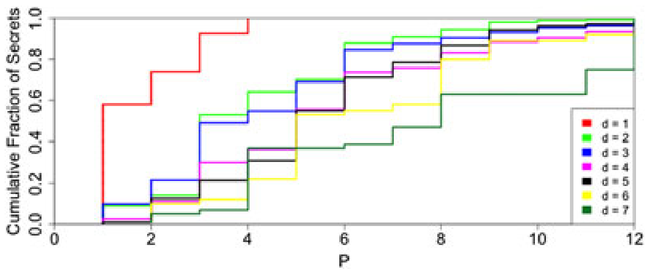

FIGURE 8 Empirical evaluation of the number P of possible key combinations in NomadiKey for a given set of touch positions and large (PIN-size) keys

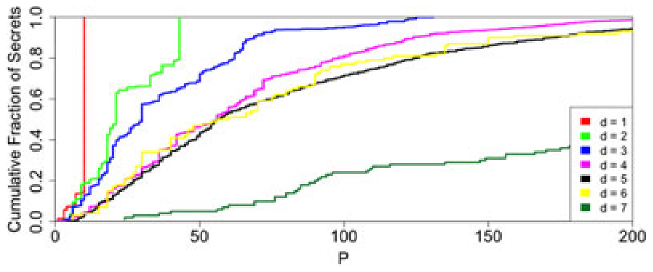

FIGURE 9 Empirical evaluation of the number *P* of possible key combinations in NomadiKey for a given set of touch positions and small (qwerty-size) keys

TABLE 4 Average number of combinations

	Extra Digits (f)		
Secret Length (n)	**1**	**2**	**3**
3	2.4	4.57	7.18
4	2.86	5.91	10.67
5	3.19	7.53	14
6	3.6	9.05	19.1

5.2 | 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 4 authentication mechanisms in an Android application. We considered PINs of 4 digits, based on average secret length of 4.5 digits found in previous work.[42]

We asked volunteers to authenticate in our developed application using each of the 4 authentication mechanisms. For each mechanism, the user was presented with 5 different random secrets. The user authenticated 3 times with each of the 5 secrets for a total of 15 authentications per mechanism. For each authentication, the user presses a button to display the keyboard and start the test. The application stores the time from the start of the test until the user enters the entire secret or makes a mistake. Experiments where the user makes a mistake are ignored in the evaluation of the authentication delay.

We used 2 devices to perform usability experiments: an LG G3 and an LG G4. Both devices have screens of 5.5″. Before starting a usability experiment with a user, we gave a brief overview of the experiment and a brief explanation of NomadiKey and NomadiKey ++. Users did not have any previous experience with either. We performed usability experiments with 20 volunteers: 10 females and 10 males. User age varied from 18 to 70 years.

We evaluated versions of NomadiKey with keyboard-size and PIN-size keys, the same sizes used in our empirical security evaluation (Figures 1 and 3). Recall that larger keys lead to better usability, while smaller keys lead to better security but lower usability. We also evaluate NomadiKey ++ with smaller keys but not larger keys. Because NomadiKey ++ is meant to be a security extension, only triggered when further security is needed. Hence, we only consider a version with the more secure smaller keys. We argue the usability loss from smaller keys and NomadiKey ++ is reasonable, given the security gain and sporadic use.

Figure 10 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 milliseconds). 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 (1) locate the first key they need to type and (2) reach each key. In the median case, users can authenticate on NomadiKey 40% faster than on random keyboards.

Figure 11 shows similar results for NomadiKey using small keys. As expected, decreasing key size results in longer authentication delays. We remark, however, that even with small keys, users can still authenticate faster on NomadiKey than on random keyboards (authentication delays are approximately 4000 milliseconds).

Figure 12 shows results for NomadiKey ++. Recall that we evaluate NomadiKey ++ with small keys only. It can be noted that authentication delays on NomadiKey ++ are much higher than authentication delays on other authentication

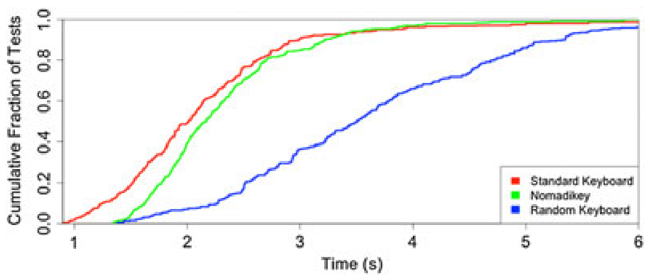

FIGURE 10 Usability of NomadiKey and PIN authentication on large keys

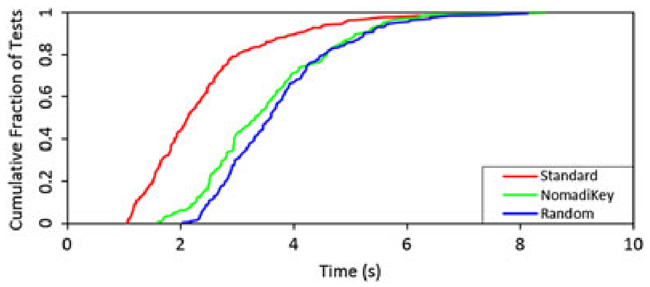

FIGURE 11 Usability of NomadiKey and PIN authentication on small keys

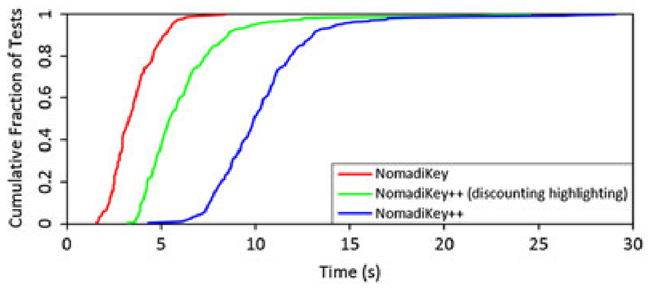

FIGURE 12 Usability of NomadiKey ++

mechanisms (Figure 11). There are mainly 2 reasons for this. First, PINs entered for NomadiKey ++ are longer (6 digits instead of 4) because users must enter their PIN and the 2 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 milliseconds, with each key being highlighted for 400 milliseconds.

We also show in Figure 12 the authentication delay of NomadiKey and NomadiKey ++ without considering the highlighting period. Ignoring the highlighting period, the usability of NomadiKey ++ is close to that of NomadiKey. We note that, while it is possible to speed up the highlighting period, doing so leads to lower usability (albeit lower authentication delays) as it becomes harder for users to determine which key was highlighted when the vibration occurred.

Finally, we argue that the combination of NomadiKey and NomadiKey ++ strikes a good trade-off between security and usability. Even for the lower usability variant of NomadiKey (ie, with smaller keys), the mechanism's usability remained at least as good as the usability of random keyboards. Being on average only 1.3 seconds 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 nontrivial security coefficient under shoulder-surfing attacks.

We believe, however, that users may consider NomadiKey cumbersome and refuse to adopt it. In particular, a significant fraction of users do not use any authentication mechanism on their smartphone[15] and most users are unwilling to use more secure mechanisms if they impact usability.[7] We claim, however, that NomadiKey looks not only viable but a powerful solution for users who are especially concerned with their privacy.

6 | 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 stronger security mechanisms if they impact usability. To address this issue, we proposed NomadiKey, a new user authentication mechanism that seeks to achieve a better trade-off between security and usability through what we call nomadic keys. We also proposed a security extension to NomadiKey, called NomadiKey ++, that can be triggered to provide better protection against attacks, especially against shoulder-surfing adversaries. We evaluated both NomadiKey and NomadiKey ++ regarding their security level and usability and compared our solutions to existing authentication mechanisms. We believe both NomadiKey and NomadiKey ++ strike an interesting trade-off between security and usability, providing better protection for the user without becoming a nuisance.

ACKNOWLEDGEMENTS

We thank Leonardo Cotta, Leandro T. C. Melo, Luiz Felipe Z. Saggioro, Frederico Martins, Antonio L. Maia Neto, and Antonio A.F. Loureiro for their valuable contributions. We also thank the IJNM reviewers for their suggestions and comments. This work is partially funded by CAPES, CNPQ, and FAPEMIG.

ORCID

Artur Souza http://orcid.org/0000-0002-6927-4275

REFERENCES

1. Ashton K. That 'Internet of Things' thing. *RFiD J*. 2009;22(7):97-114.
2. Stergiou C, Psannis KE. Recent advances delivered by mobile cloud computing and Internet of Things for big data applications: a survey. *Int J Network Manage*. 2017;27(3).
3. Kaur R, Kaur N, Sood SK. Security in IoT network based on stochastic game net model. *Int J Network Manage*. 2017;27(4):e1975.
4. Neto ALM, Souza AL, Cunha I, et al. AoT: authentication and access control for the entire iot device life-cycle. In: Sensys. Stanford, CA, USA; 2016.
5. Wang D, Lo D, Bhimani J, Sugiura K. Anycontrol–IoT based home appliances monitoring and controlling. In: COMPSAC. Taichung, Taiwan; 2015:487-492.
6. Clarke NL, Furnell SM. Authentication of users on mobile telephones—a survey of attitudes and practices. *Comput Secur*. 2005;24(7):519-527.
7. Arif AS, Mazalek A. A tap and gesture hybrid method for authenticating smartphone users. In: MobileHCI. Munich, German; 2013: 486-491.
8. Yue Q, Ling Z, Fu X, Liu B, Ren K, Zhao W. Blind Recognition of touched keys: attack and countermeasures. *CoRR*. 2014;abs/1403.4829.
9. Fischer IT, Kuo C, Huang L, Frank M. Short paper: smartphones: not smart enough? In: SPSM. Raleigh, North Carolina, USA; 2012:27-32.
10. Wiedenbeck S, Waters J, Sobrado L, Birget J-C. Design and evaluation of a shoulder-surfing resistant graphical password scheme. In: Avi. Venezia, Italy; 2006:177-184.
11. Aviv AJ, Gibson K, Mossop E, Blaze M, Smith JM. Smudge attacks on smartphone touch screens. In: WOOT. Washington, DC; 2010.
12. Shukla D, Kumar R, Serwadda A, Phoha VV. Beware, your hands reveal your secrets! In: CCS. Scottsdale, Arizona, USA; 2014:904-917.
13. Maggi F, Volpatto A, Gasparini S, Boracchi G, Zanero S. Poster: fast, automatic iPhone shoulder surfing. In: CCS. Chicago, Illinois, USA; 2011:805-808.
14. Cotta L, Fernandes AL, Melo LTC, et al. NomadiKey: user authentication for smart devices based on nomadic keys. In: Icc. Kuala Lumpur, Malaysia; 2016:1-6.
15. Andriotis P, Tryfonas T, Yu Z. Breaking the android pattern lock screen with neural networks and smudge attacks. In: WiSec. Oxford, United Kingdom; 2014.
16. Bojinov H, Boneh D. Mobile token-based authentication on a budget. In: HotMobile. Phoenix, Arizona; 2011:14-19.
17. Furnell S, Clarke N, Karatzouni S. Beyond the PIN: enhancing user authentication for mobile devices. *Comput Fraud Secur*. 2008;2008(8):12-17.
18. De Luca A, Hang A, Brudy F, Lindner C, Hussmann H. Touch me once and I know it's you!: implicit authentication based on touch screen patterns. In: CHI; 2012.
19. Jakobsson M, Shi E, Golle P, Chow R. Implicit authentication for mobile devices. In: Hotsec. Montreal, Canada; 2009:9-9.
20. Liu J, Wang Z, Zhong L, Wickramasuriya J, Vasudevan V. uWave: accelerometer-based personalized gesture recognition and its applications. In: PerCom; 2009;5(6):657-675.
21. Mock K, Hoanca B, Weaver J, Milton M. Real-time continuous iris recognition for authentication using an eye tracker. In: CCS. Raleigh, North Carolina, USA; 2012:1007-1009.

22. Pan S, Chen A, Zhang P. Securitas: user identification through RGB-NIR camera pair on mobile devices. In: SPSM. Porto, Portugal; 2013:168-185.
23. Wang H, Lymberopoulos D, Liu J. Sensor-based user authentication. In: EWSN. Scottsdale, Arizona, USA; 2015:51-62.
24. Zhang L, Tan S, Yang J, Chen Y. VoiceLive: a phoneme localization based liveness detection for voice authentication on smartphones. In: CCS. Vienna, Austria; 2016:1080-1091.
25. Meng W, Wong DS, Furnell S, Zhou J. Surveying the development of biometric user authentication on mobile phones. *IEEE Commun Surv Tutorials*. 2015;17(3):1268-1293.
26. Chen Y, Sun J, Zhang R, Zhang Y. Your song your way: rhythm-based two-factor authentication for multi-touch mobile devices. In: INFOCOM. Kowloon, Hong Kong; 2015:2686-2694.
27. Haque SMT, Wright M, Scielzo S. Passwords and interfaces: towards creating stronger passwords by using mobile phone handsets. In: SPSM. Berlin, Germany; 2013:105-110.
28. Jermyn I, Mayer A, Monrose F, Reiter MK, Rubin AD. The design and analysis of graphical passwords. In: USENIX Security. Monterey, California, USA; 1999.
29. Wiedenbeck S, Waters J, Sobrado L, Birget J-C. Design and evaluation of a shoulder-surfing resistant graphical password scheme. In: AVI. Venezia, Italy; 2006:105-110.
30. Krombholz K, Hupperich T, Holz T. Use the force: evaluating force-sensitive authentication for mobile devices. In: SOUPS'16. Austin, Texas; 2016.
31. Egelman S, Jain S, Portnoff RS, Liao K, Consolvo S, Wagner D. Are you ready to lock? In: CCS. Scottsdale, Arizona, USA; 2014:750-761.
32. Dell'Amico M, Michiardi P, Roudier Y. Password strength: an empirical analysis. In: INFOCOM. San Diego, CA, USA; 2010:1-9.
33. Dunphy P, Yan J. Do background images improve draw a secret graphical passwords? In: CCS. Alexandria, Virginia, USA; 2007:36-47.
34. Uellenbeck S, Dürmuth M, Wolf C, Holz T. Quantifying the security of graphical passwords: the case of android unlock patterns. In: CCS. Berlin, Germany; 2013:161-172.
35. Zhang Y, Xia P, Luo J, Ling Z, Liu B, Fu X. Fingerprint attack against touch-enabled devices. In: SPSM. Raleigh, North Carolina, USA; 2012:57-68.
36. Yue Q, Ling Z, Fu X, Liu B, Ren K, Zhao W. Blind recognition of touched keys on mobile devices. In: CCS. Scottsdale, Arizona, USA; 2014:1403-1414.
37. Raguram R, White AM, Goswami D, Monrose F, Frahm J-M. iSpy: automatic reconstruction of typed input from compromising reflections. In: CCS; 2011:527-536.
38. Schaub F, Deyhle R, Weber M. Password entry usability and shoulder surfing susceptibility on different smartphone platforms. In: MUM. Chicago, Illinois, USA; 2012:527-236.
39. Simon L, Anderson R. PIN skimmer: inferring PINs through the camera and microphone. In: SPSM. Berlin, Germany; 2013:67-78.
40. Spreitzer R. PIN skimming: exploiting the ambient-light sensor in mobile devices. In: SPSM. Scottsdale, Arizona, USA; 2014:51-62.
41. Rosen KH. Discrete mathematics and its applications. *AMC*. 2007;10(12):824.
42. Harbach M, von Zezschwitz E, Fichtner A, Luca AD, Smith M. It's a hard lock life: a field study of smartphone (un)locking behavior and risk perception. In: SOUPS; 2014.

Artur Souza received his BSc in Computer Science from the Federal University of Minas Gerais (UFMG), Brazil, in 2016. He is currently a PhD student in Computer Science at the Federal University of Minas Gerais. He has coauthored papers published in SenSys and IEEE ICC and is inventor of a patent for an authentication scheme for IoT. His research interests include security, applied cryptography, Internet of Things, and vehicular-to-everything communication (V2X).

Ítalo Cunha is an assistant professor at the Computer Science Department at UFMG, Brazil since 2012. He developed his PhD research at Technicolor Research and Innovation Paris and graduated from UPMC Sorbonne Universités in 2011. His research focuses on improving network performance and reliability. His contributions provide better visibility on Internet topology and routing dynamics, help network operators troubleshoot failures and performance problems, and empower other researchers. Ítalo has served on the technical committee of flagship networking conferences such as ACM IMC and ACM SIGCOMM.

Leonardo B Oliveira has been awarded the Microsoft Research PhD Fellowship Award, the Intel Strategic Research Alliance Award, and the IEEE Young Professional Award. He led projects funded by companies like Intel Labs and LG Electronics Mobile Research. He published over a hundred scientific papers in refereed venues like IEEE/ACM IPSN and ACM Sensys and he is the inventor of an authentication scheme for IoT (USPTO No. 62287832). His Google citation count and h-index are approximately 2000 and 20, respectively. He is a member of the Technical Committee of Identity Management (CT-GId) of the Brazilian National Research and Educational Network (RNP), holds a position in the Advisory Board of the Special Interest Group on Information and Computer System Security (CESeg) of the Brazilian Computer Society, and served as General and TPC Chair of the Brazilian Symposium on Security (SBSeg) in 2014 and 2016, respectively. His research interests include security and applied cryptography for IoT/Cyber-Physical Systems.

How to cite this article: Souza A, Cunha Í, B Oliveira L. NomadiKey: User authentication for smart devices based on nomadic keys. *Int J Network Mgmt*. 2018;28:e1998. https://doi.org/10.1002/nem.1998