

# Why Mobile Security is not Like Traditional Security

*Invited Paper*

Markus Jakobsson\*

PayPal  
San Jose, CA

**Abstract.** We argue that traditional security solutions do not necessarily work for mobile environments, but that suitable solutions can and must be developed to address the needs of mobile platforms. To support this, we describe how the differences are likely to affect security and describe mobile-friendly solutions for CAPTCHAs, user authentication, and malware protection.

## 1 Introduction

It is a bit worrisome that consumers do not think smartphones are computers. For example, only 32% of respondents in a 2010 survey on Amazon Mechanical Turk believe that that phones could get infected by malware. However, in many ways, it is a much greater danger that many computer scientists still think that smartphones *are* computers – and nothing *but* computers. This paper argues that the many differences in form factor and functionality set handsets distinctly apart from traditional computers in terms of security – both in terms of the threats to these devices, and the countermeasures that must be developed and applied to maintain security.

The most notable differences between handsets and traditional computers (from here on simply called *computers*) are the much smaller screens and keyboards of handsets. This has a security impact. For example, the smaller screens mean that webpage address bars are often automatically hidden to make room for other content; this has been pointed out to be easy to take advantage of by phishers [6]. Similarly, the smaller keyboards make users more likely to use password managers or short passwords. It makes service providers settle for PINs instead of passwords and to implement remember-me features to minimize friction. All of these developments can have negative effects on security unless we pay careful attention to how security features are designed and managed.

Moreover, the processing limitations and the severe battery power limitations of handsets make some security technologies impractical. Many consumers decide not to use Anti-Virus products because of the resulting slow-down of their

---

\* Some of the work was performed while the author was with Palo Alto Research Labs; FatSkunk; and RavenWhite.

computers. These effects will be even more dramatic on handsets. Furthermore, the battery constraints may make the traditional anti-virus paradigm meaningless on handsets: Hardly anybody would be willing to use a product that might drain the batteries of the phone within an hour, say. While this is not the current situation for existing mobile anti-virus solutions, we believe that this is simply due to the relative absence of mobile threats, since the cost of scanning grows with the number of things scanned for. As mobile payments take off [16] and the number of smartphones surpass the number of Windows machines in the world within a matter of years [7], we believe that malware authors will turn to mobile malware in earnest. This will drive up the costs of managing the countermeasures of the traditional types to levels where consumers opt out.

The vulnerabilities are not only results of technical differences, but also of *behavioral* aspects. People use phones differently from how they use computers [15]. For one thing, phones are more strongly associated with social activities than computers typically are. We are not aware of any studies in which it is determined whether this may affect security, but have anecdotal evidence that it might. For example, a study on phishing [8] showed that close to 75% of users entered their credentials on a simulated phishing site when prompted to do so by a friend – or rather, by an email believed to come from a friend. This is a power of magnitude greater than the portion of users who did it when they were prompted by a stranger.

While smartphones suffer from many security drawbacks in comparison to computers, this is not a reason for us to give up and adopt a very limited scope of use of these devices. While we do not think that traditional security solutions can always be ported to work on handsets, we argue that the obstacles and threats we have outlined above can be overcome, and that sometimes, the new solutions may be able to take advantage of features that computers do not offer [14].

To support our claim that mobile security is both *different* and *possible*, we describe an array of mobile security solutions. We begin in section 2 by describing security features designed with restricted keyboards in mind. In particular, this section describes solutions for CAPTCHAs and authentication methods that are well suited for mobile devices. Then, in section 3, we detail novel security approaches designed for devices with limited battery resources – in particular, anti-malware approaches that are well suited for handsets.

## 2 Dealing with Restricted Keyboards

Text entry on handsets is time-consuming and error-prone. Consumers are only *slightly* less frustrated by entry of text and passwords on handsets than they are of slow web connections on such devices, and much more annoyed with all of these than lack of coverage and poor voice quality [14]. This is not only a user experience problem, but also a security issue – since users typically attempt to circumvent inconveniences, causing security vulnerabilities in the process. We will review a collection of solutions that demonstrate the existence of “mobile-friendly” CAPTCHAs and authentication techniques.

## 2.1 Clickable CAPTCHAs

To make CAPTCHAs practical for handsets, Chow et al. [3] developed a *clickable* CAPTCHA. This can be based on any underlying textual CAPTCHAs – one example is shown figure 1. There, we created 12 Google CAPTCHAs and tiled them in a 3-by-4 grid. Of the 12 CAPTCHAs, 3 represent *real* English words (chosen at random from a dictionary) whereas the remaining 9 do *not* correspond to any English word. The 3 CAPTCHAs containing English words are placed in random locations in the 3-by-4 grid.



**Fig. 1.** Example of a clickable CAPTCHA. The user’s task is to identify the three valid English words (in this example: *monster*, *grass* and *nation*). On a handset with touch screen, the user would tap on the three English words. On one without touch screen, she would touch the corresponding key – the matrix of CAPTCHAs maps directly to the 12 keys on the handset.

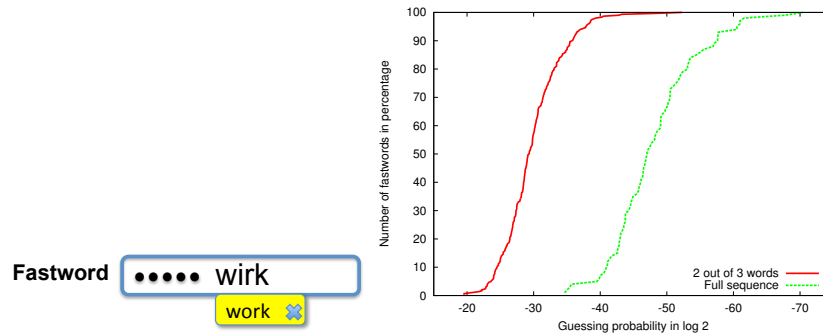
To solve this clickable CAPTCHA, the user must click on the 3 cells which contain English words, and only those. The correct solution requires exactly 3 clicks. Any click on another CAPTCHA cell invalidates the solution. Note that an English-speaking user can trivially solve this clickable CAPTCHA: The user scans the 12 cells, recognizes the 3 cells which contain English words and clicks on them. A computer, on the other hand, that cannot decode the individual CAPTCHAs can also not solve the clickable CAPTCHA.

To further prevent computers from automatically telling apart English words from non-English words, the sequences of letters which are not English words can be generated according to a Markov process which models the distribution of bigrams and trigrams in English. Alternatively, non-English words could be generated with simple transformations of English words.

## 2.2 Fastwords Instead of Passwords

While auto-correction and auto-completion are very common, they only work for *text*. Neither works for password entry. This is due to the fact that good passwords are much like poorly spelled words, and that error-correction techniques help *remove* poor spelling, at the same time as the ability to decompose into components makes it easy to assess the strength of credentials.

A *fastword* [10] is a sequence of dictionary words, separated by spaces. Since the entry of fastwords benefits from error-correction methods, they can be entered about as fast on handsets as passwords can be entered on traditional keyboards. A portion of a sample user interface is shown in Figure 2, along with a graph showing the approximate security of the approach.



**Fig. 2.** The left part of the figure shows what the user may see when entering a fastword. The first word has been replaced by stars, and the second word is shown with an autocorrect suggestion. The use of auto-correct and auto-complete allows users to type faster and with less precision. To accept a suggestion, the user simply continues writing. The “full sequence” graph on the right shows the strengths of partial fastwords of one hundred subjects, using fastwords consisting of three consecutive dictionary words. It also shows the security when two out of the three keywords are entered and the third is left out. In comparison, traditional passwords are believed to have approximately 18 bits of security [2].

Let’s consider an example: A particular user may choose the sequence “frog work flat”, which might correspond to a mnemonic of “I ran over a frog on my way to work, and now I have a flat frog under the tire.” Our example system would accept this as a strong fastword, given that the frequencies of the product of the three words in regular language is  $2^{-42.3}$  and the 3-gram frequency [20] is  $2^{-49.5}$ . (This corresponds to more than twice the bit strength of common passwords [2], and largely matches the security of *completely random* 8-character passwords [4].)

On the other hand, a user who enters the fastword “I love you honey” would be told to think of a stronger fastword. This is spite of the fact that the prod-

uct of frequencies is  $2^{-43.7}$ , which suggests a *stronger* credential than our first example. The reason is that the 4-gram frequency is only  $2^{-25.8}$ . Bit strength distributions of fastwords are shown in figure 2. It can be seen that this new approach tolerates partial omissions without detrimental security results; it can also be used in combination with dictation software instead of keyboard-entry techniques. See [10] for more details.

### 2.3 Bootstrapping PINs from Passwords

Many financial institutions are gravitating towards PINs when passwords are impractical. While PINs offer improved convenience to users, having to force millions of existing users to create PINs is not desirable at all.

We propose *bootstrapping* the generation of PINs from passwords to create an automated onboarding of an overwhelming majority of users [13]. A possible user interface is shown in figure 3.



**Fig. 3.** The figure shows the user interface. To the left is an image of what the user might see when arriving at the authentication stage; in some contexts, the user name may be auto-filled or obtained from a drop-down menu. The image to the right shows what she would see after tapping on the PIN window. To log in, the the consumer would simply enter the first four characters of her password, using the numeric keypad. If a password character is “2”, “A”, “a”, “B”, “b”, “C” or “c”, then the user presses the 2-button – we are not case sensitive. A password starting with “Blu2” would correspond top the PIN 2852.

One can derive PINs from already established passwords when these are available in cleartext on the backend. This requires no user involvement. The user would, in fact, not even be aware of the PIN creation until she is told that she *has* a PIN, and should use it to log in. The PIN is set as the first four characters of the password, mapped to a numeric keypad. This mapping is analogous to how alphanumeric phone numbers are mapped to a 10-button keypad – like 1-800 CALL ATT *becomes* 1-800 225 5288. Similarly, the password “Blu2thrules” becomes the PIN “2582”.

By analyzing thousands of real-life passwords from dropboxes, we can determine the associated PINs that would have been derived. We can also compute the resulting entropy, which is close to 10 bits. (This is slightly more secure than traditional PINs, since many users make weak selections [1].) It is also the amount of information an adversary gains about a password if he compromises the corresponding PIN. This leaves approximately ten bits of conditional entropy in the password [2]. For most practical purposes, this is sufficient, given that it is straightforward for the backend to detect incorrect passwords that map to correct PINs – a near-certain sign of compromise.

### 3 Dealing with Limited Battery Resources

Defense of mobile phones against malware is complicated by the fact that these devices suffer severe power restrictions, which places limitations on what type of detection can be performed. It is not evident that the traditional anti-virus paradigm – with its use of signature detection and behavioral detection – is well suited for mobile platforms. To deal with this problem, a method referred to as *software-based attestation* [18, 19, 11, 12, 5] has been developed.

Software-based attestation can be performed by making sure that there are no unwanted processes running, after which all memory contents can be inspected. As a first step, an external verifier wishes to establish that the only process running on a target machine is an unmodified version of the audit process. If this verification succeeds, then the external verifier knows that there is no *active* malware. Note that the detection process only has to be run occasionally, since it offers *retroactive* detection of anything that has infected the device (and remained active.) Therefore, the power consumption is reduced in comparison with that of traditional anti-virus software, which essentially has to run all the time to be effective.

As soon as it has been determined that a device has no active malware, then its entire secondary storage can be screened or copied to a cloud-based storage facility, where it can be screened [17]. This screening will detect any known *inactive* malware, and can identify infections based on propagation patterns [9]. As soon as a new piece of malware has been identified anywhere in the system, everybody’s cloud-based repositories can be screened to see if they were infected in the past. This provides a *second* type of retroactive detection ability. It is also possible to side-step the screening process and simply perform security sensitive

processes – such as initiating payments – once it has been determined that there is no active malware.

One can verify that there are no unwanted processes running by performing so-called *memory-printing* [11]. This is a process that is configured to require all of *free* RAM to complete its computation in the expected amount of time – or rather, all RAM that *should be* free after all active processes have been instructed to page out. Memory printing has the property that any reduction of space leads to measurable slowdowns of the computation, which is detected by the external verifier. Moreover, it is performed in a manner that results in a notable slowdown if the process is modified to sometimes access secondary storage instead of RAM. *Therefore, if a malware agent is active – i.e., remains in RAM – then the process will be slowed down.*

## 4 Conclusion

We have argued that mobile security cannot always be achieved by porting existing security solutions to handsets and other mobile devices. Instead, new solutions have to be designed to take the constraints of the new devices into consideration. These solutions may depend on and relate to the security properties of some underlying constructions – as in the clickable CAPTCHA case – or may be designed from scratch, potentially using specific features of the host device. One example of the latter is the *fastword* proposal, where the solution uses error-correction techniques developed for handsets; another example is the malware-detection construction we described, in which the security of the solution depends on hardware specifics, such as access times for various types of components.

## References

1. S. Berry. One in five use birthday as PIN number, Daily Telegraph, 27 Oct 2010.
2. W. E. Burr, D. F. Dodson, R. A. Perlner, W. T. Polk, S. Gupta, E. A. Nabbus, C. M. Gutierrez, and J. M. Turner. Special Publication 800-63-1 Electronic Authentication Guideline, 2008.
3. R. Chow, P. Golle, M. Jakobsson, L. Wang, and X. Wang. Making CAPTCHAs clickable. In *HotMobile '08: Proceedings of the 9th workshop on Mobile computing systems and applications*, pages 91–94, New York, NY, USA, 2008. ACM.
4. D. A. F. Florêncio and C. Herley. A large-scale study of web password habits. In *WWW*, pages 657–666, 2007.
5. J. A. Garay and L. Huelsbergen. Software integrity protection using timed executable agents. In *ASIACCS '06: Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, pages 189–200, New York, NY, USA, 2006. ACM.
6. D. Goodin. Researcher warns of iPhone phishing peril, [http://www.theregister.co.uk/2010/11/29/iphone\\_phishing\\_threat/](http://www.theregister.co.uk/2010/11/29/iphone_phishing_threat/).
7. S. Havlin. Phone infections. *Science*, 324(5930):1023–1024, 2009.

8. T. N. Jagatic, N. A. Johnson, M. Jakobsson, and F. Menczer. Social phishing. *Commun. ACM*, 50(10):94–100, 2007.
9. M. Jakobsson. A central nervous system for automatically detecting malware, [blogs.parc.com/blog/2009/09/a-central-nervous-system-for-automatically-detecting-malware/](http://blogs.parc.com/blog/2009/09/a-central-nervous-system-for-automatically-detecting-malware/), September, 2009.
10. M. Jakobsson and R. Akavipat. Rethinking Passwords to Adapt to Constrained Keyboards, Draft report, available at [www.fastword.me](http://www.fastword.me), 2010.
11. M. Jakobsson and K.-A. Johansson. Retroactive detection of malware with applications to mobile platforms. In *ACM HotSec 10*, 2010.
12. M. Jakobsson and K.-A. Johansson. Practical and Secure Software-Based Attestation. In *Proceedings of LightSec*, 2011.
13. M. Jakobsson and D. Liu. Bootstrapping Mobile PINs Using Passwords, Draft report, available by request, 2010.
14. M. Jakobsson, E. Shi, P. Golle, and R. Chow. Implicit authentication for mobile devices. In *HotSec'09: Proceedings of the 4th USENIX conference on Hot topics in security*, pages 9–9, Berkeley, CA, USA, 2009. USENIX Association.
15. T. Kee. Study: Smartphone users get an 'F' when it comes to security, August 17, 2009.
16. J. Mont. Mobile Payments Poised to Explode in 2011, <http://www.thestreet.com/story/10936307/mobile-payments-poised-to-explode-in-2011.html>.
17. J. Oberheide, E. Cooke, and F. Jahanian. CloudAV: N-Version Antivirus in the Network Cloud. In *Proceedings of the 17th USENIX Security Symposium*, San Jose, CA, July 2008.
18. A. Seshadri, M. Luk, E. Shi, A. Perrig, L. van Doorn, and P. Khosla. Pioneer: verifying code integrity and enforcing untampered code execution on legacy systems. In *SOSP '05: Proceedings of the twentieth ACM symposium on Operating systems principles*, pages 1–16, New York, NY, USA, 2005. ACM Press.
19. E. Shi, A. Perrig, and L. V. Doorn. Bind: A fine-grained attestation service for secure distributed systems. In *SP '05: Proceedings of the 2005 IEEE Symposium on Security and Privacy*, pages 154–168, Washington, DC, USA, 2005. IEEE Computer Society.
20. K. Wang, C. Thrasher, E. Viegas, X. Li, and B.-J. Hsu. An overview of Microsoft web N-gram corpus and applications. In *Proceedings of the NAACL HLT 2010 Demonstration Session*, HLT '10, pages 45–48, Morristown, NJ, USA, 2010. Association for Computational Linguistics.