When Users Cannot Verify Digital Signatures: On the Difficulties of Securing Mobile Devices

Rene Mayrhofer
University of Applied Sciences Upper Austria

Softwarepark 11, 4232 Hagenberg, Austria
Email: rene.mayrhofer@fh-hagenberg.at

***Abstract*—Mobile devices such as smart phones have become one of the preferred means of accessing digital services, both for consuming and creating content. Unfortunately, securing such mobile devices is inherently difficult for a number of reasons. In this paper, we systematically analyze the technical issues of securing mobile device platforms against different threats and discuss a resulting and currently unsolved problem: how to create an end-to-end secure channel between the digital service (e.g. a secure wallet application on an embedded smart card or an infrastructure service connected over wireless media) and the user. Although the problem has been known for years and technical approaches start appearing in products, the user interaction aspects have remained unsolved. We discuss the reasons for this difficulty and suggest potential approaches to create human-verifiable secure communication with components or services within partially untrusted devices.**

***Keywords*-mobile device security; user authentication; secure channel; virtualization; embedded smart card**

I. Introduction

Mobile devices in their various physical incarnations such as smart phones, wrist watches, glasses, or other forms of wearable computing are replacing traditional clients for accessing information services. They already store highly sensitive and private data and are involved in processing monetary transactions; in the near future, they will most likely also process medical data and represent the user in even more situations by acting as their digital proxy, e.g. for digital identification.

With the transition from stationary to small mobile devices, users gain mobility, sensing, context awareness, and integration, but lose extensibility based on well-known hardware interfaces. In the past, these interfaces (e.g. USB, PCMCIA, SD, or PCI) have been used on desktops and laptops to connect trusted hardware components (e.g. smart card readers with integrated keypad, USB mass storage devices with fingerprint readers, etc.).

From a security point of view, these integrated security devices can provide trusted services such as key storage or cryptographic computations even under the assumption of a (partially) untrusted platform as the rest of the computing system. In fact, class-2 smart card readers with integrated pinpads are a required hardware component in many national laws on so-called 'advanced' electronic signatures with 'qualified' certificates [1] to prevent the well-known problem of malware logging – and invisibly using – the PIN codes that are required to unlock private keys stored on users' smart cards. On the one hand, current mobile devices do not easily offer the required interfaces to connect such trusted hardware components, nor do we expect end-users to desire carrying a second device to use only for security relevant input or output. On the other hand, the software stack of mobile devices like smart phones or watches cannot be fully trusted and will not be able to reach sufficiently high levels of certification (such as CC EAL 4+) because of the inherent complexity of the combined set of kernel, libraries, system binaries, runtime execution environments, and installed applications that almost necessarily leads to security relevant bugs (e.g. [2], [3]). We therefore face the problem of providing comparable – or for some future scenarios even better – security on hardware/software platforms that cannot be fully trusted.

Traditionally, cryptographic protocols have been applied to secure communication over untrusted channels, e.g. by using TLS/SSL to connect to a web server over wireless channels. The problem in this case is that end users cannot directly execute cryptographic protocols without any help: they cannot directly verify digital signatures or perform a decryption operation for a modern cipher. This leads to the central open issue in mobile device security: *how can users trust their communication with an embedded secure hardware component* (e.g. a smart card embedded within their smart phone) *when they cannot realistically trust the normal user interface of their device* (which is built upon many layers of complex software components)?

II. Security Threats for Mobile Devices

Before discussing specific technical approaches, we analyze the security threats that current mobile devices and their users face. We illustrate the threats with the following use cases in mind: physical access control (smart phone as wireless key), mobile payment (smart phone as credit/debit card or with locally stored digital coins such as Bitcoins [4]),

the smart phone as identity document (virtual passport, driving license, etc.), and the smart phone as access terminal for remote data (e.g. company email, ERP, etc.). Note that in contrast to Egners et al. [5] who distinguish between owner threats, platform threats, threats to other users, and mobile network operator threats, we focus on threats to the end user and their data (owner threats) as the most challenging class. In contrast to La Polla et al. [6], we do not assume mobile phones to be different from desktop/laptop type systems in terms of CPU and/or memory capability, but focus on usability and the context of use as the main distinguishing factors.

All of the above use cases assume the basic security requirements, namely that user data should remain confidential (C), that user data integrity (I) should be protected, and that this data and services need to remain accessible (A) to authorized users. We can easily derive that the corresponding threats are leaking private data, modifying user data, or rendering the device or its communication inaccessible. These abstract threats translate to more detailed threats on different aspects of the mobile device and its use:

- *Physical access* (loss, theft, borrowing): Devices under the physical control of illegitimate third parties should still protect the private data (e.g. digital coins or the identity documents) of their legitimate owners. This is also referred to as a "malicious user" threat and is currently mostly addressed for loss or theft by on-device encryption (cf. next section) but largely open for borrowing of mobile devices [7]. A sub-class is the illegitimate use of devices by their owners to e.g. circumvent copy protection schemes (which falls under platform threats in the classification in [5])
- *Communication*: As in most related work, we assume a Dolev-Yao attacker on all wireless channels: an adversary can eavesdrop, delay, drop, replay, spoof, and modify messages and masquerade as any sender. Additionally, relay attacks e.g. on the embedded smart card may not directly attack the cryptographic protocol, but still be able to exploit communication [8].
- *Platform*: Due to standard security issues in operating systems, libraries, or applications, the platform itself can be attacked with the aim of violating any of the security assumptions, e.g. to read private user data, modify data, or perform a denial-of-service attack. We can further distinguish between:
  - External attacks exploiting any of the multitude of wireless interfaces (e.g. WLAN, Bluetooth, NFC, or cellular network such as UMTS) or on protocols of upper layers (e.g. HTTP, HTML parsers, etc.) may be able to directly access private data or lead to remote code execution.
  - Internal attacks may be performed by installing malicious apps to either read sensitive data based on standard application permissions (granted by inattentive users or not properly enforced by the platform) or might exploit further privilege escalation issues [3] to gain full access to the platform with arbitrary permissions.
- *User interaction*: The most difficult class of threats concerns user interaction: installed applications (malware) may try to fake the look and feel of other apps or platform components, display erroneous or fake data, capture user input with key/touch logging, and mislead or confuse the user into making wrong decisions. These threats are difficult to address because they not only involve technical, but also psychological and potentially social aspects, e.g. by exploiting peer pressure or pretended authority in social engineering attacks. We discuss the major differences to laptop/desktop systems in more detail below (see section IV).

III. Technical Approaches

From a technical point of view, addressing the threats to end-user data and services requires securing all involved layers of current mobile device platforms:

1) *Hardware* executes all firm- and software, and therefore a minimal set of trusted hardware is required as the root of the trust chain. Although TPMs have been offering secure key storage and code execution, monitoring of the boot process, and extended protocols for remote attestation (allowing a device to provide proof to a remote service that it is only executing certified software) for laptops, security issues were found [9] and they have not been widely integrated into off-the-shelf mobile phones or similar devices so far. Instead, *secure elements* (SEs) are starting to appear as part of the NFC hardware stack in the form of embedded smart cards implementing the JavaCard standard.
   SEs can be utilized for secure key storage and cryptographic operations, but cannot (at the time of this writing) support monitoring the boot process to provide a trust anchor for code executed on the main application processor (AP). Therefore, implementing secure boot of the main mobile operating system (the interface to the next layer) still requires additional hardware support to verify the boot loader code. In combination with hardware compartmentalization features such as the ARM TrustZone, SEs can be used as a basis for secure mobile devices. Note that we do not assume the mass storage (e.g. NAND flash) to be secure on the hardware layer, but that – with physical access to the device – all mass storage can be read or modified.
2) *Platform* support includes both kernel (executing without hardware restrictions on the AP) and user space (restricted by the AP) components, which should be written with secure coding practices to prevent typical

classes of code-level security vulnerabilities (such as buffer overflows, missing input validation, etc.). Additionally, the platform should isolate applications against each other using sandboxing and verify executed code based on code signing. However, realistically we always have to assume security relevant bugs in the code due to the inherent complexity of current mobile operating systems (cf. [2], [3]). We therefore suggest that two specific security measures should be added on the platform layer:

- On-device memory encryption (as implemented e.g. by Android and iOS) is an effective safeguard against malicious user threats. Even under direct physical control, sensitive user data cannot be decrypted by attackers as long as a sufficiently long cryptographic key has been used and that key is not leaked by the platform. The interesting challenge is how to store the key; either end-users need to enter the key (or a password from which the key can be derived) at bootup with the obvious trade-off between usability and security, or it needs to be stored within a secure hardware component such as an embedded SE and unlocked with secure user authentication (see below).
- To improve the security of current sandboxing approaches, virtualization can be used to address attacks against the platform by keeping the required trusted code base as small as possible (see e.g. [10] for a brief introduction based on the older Symbian OS architecture). Only the virtualization layer – often called *hypervisor* – needs to become part of the trusted base, as the main operating system (e.g. Android) cannot realistically be assumed to be secure. ARM TrustZone supports splitting code running on the AP into trusted and untrusted parts, and starting with the Cortex-A15 generation, supports full hardware virtualization to assist existing hypervisors (e.g. Xen or KVM). An important question concerning sandboxing and virtualization that has not yet been fully addressed is that of granularity. We suggest that application sandboxing, multiple users, zones (cf. 'faces' in [11]), and virtualization of full OS instances are complementary and will be used in parallel for solving different use cases (such as phone sharing vs. solving the bring-your-own-device problem).

3) *Apps* should ideally also be implemented with secure coding guidelines to avoid data leaks on the application level. However, with the assumptions of third-party app markets and low entrance barriers for app developers, we have to assume apps to be insecure and therefore require the platform to protect itself from malicious apps (see above) and focus on securing communication between apps, infrastructure services, and users (see below). For an extensive survey on previously suggested approaches for proactively or reactively detecting security breaches by malicious or faulty applications, we refer to [6].
4) *Communication channels* can be secured effectively with well-known cryptographic protocols such as TLS or IPsec to prevent eavesdropping or manipulation of user data by any parties in-transit. This includes attackers on the (wireless) transfer channels as well as malicious apps executed on the phone. With regards to confidentiality and integrity of communication, the problem is mostly solved (as long as the cryptographic primitives remain unbroken and keys are not leaked). The remaining challenge is authenticating those devices or services that are communicating with each other, which requires the user in the loop [12] to verify bootstrapping of cryptographic protocols and to prevent man-in-the-middle attacks under the assumed Dolev-Yao attacker.
5) *User interaction* involves two aspects:
   - User authentication is required to ensure that the correct user (typically the owner) is interacting with the device and as additional part to safeguard against malicious user attacks. In addition to standard password/PIN entry, (static and/or dynamic) biometric features may offer a better trade-off between usability and security, but can be difficult to secure against recorded input data [13]. To further alleviate that trade-off, Riva et al. propose a more dynamic determination of authentication events [14] that seems a a good step forwards.
   - After successful authentication, the user interface itself has to be made secure so that users can reasonably assume that the data they view and enter is not leaked or modified by other application code running in parallel on the same device.

Figure 1 summarizes the existing and currently missing technical parts required to create a secure personal mobile device.

IV. Usability Issues

Although the most prevalent classes of mobile devices currently used for accessing information services (smart phones and tablets) no longer need to be assumed to have limited resources in terms of CPU or memory capabilities (in contrast to the assumptions in [6]), there are significant differences from a usability point of view that severely impact their security design:

- Mobile devices are assumed to be personal devices, even more so than laptops. Especially smart phones are hardly shared with others, as they often act as a personal assistant in terms of routines of (professional

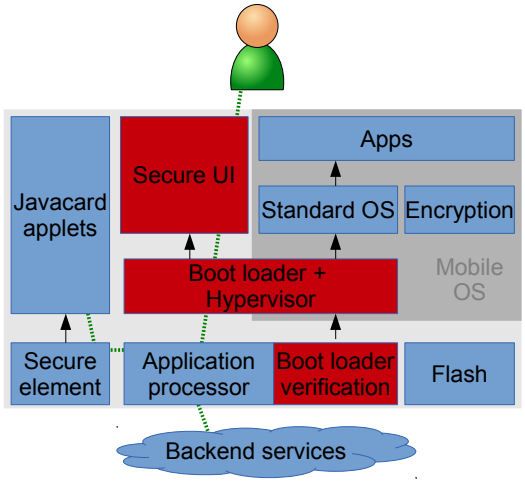

Figure 1. Overview of technical components: trust chain indicated with arrows, currently missing parts in red, end-to-end secure channel marked with dotted line

and personal) daily life. This assumption creates an interesting ambivalence in terms of usable security: On the one hand, these mobile devices are often used to keep highly personal and private information, such as messages, pictures, videos, calendar entries, contacts, or location and itinerary data. On the other hand, the same assumption of being a highly personal device often leads to a puzzling neglect of security best practices, such as using long passwords for authenticating to the device or the use of on-device encryption. Many anecdotal and semi-formal user studies show that only a minority of smart phone users enables any form of regular authentication at all (and often only if forced by security policies set by administrators of some organization). Lacking hard psychological evidence, we can only assume three underlying causes for this obvious conflict:

a) The assumption of a smart phone being a highly personal device carries the implication that the device is under the sole control of the respective user, because it is carried close to the body. Although this may be true most of the time, recent statistical data on mobile phone theft shows that it is a dangerous assumption from a security point of view. b) Following security best practices is bothersome (slow and error-prone) with the limited user input/output capabilities of current mobile devices. c) Users are unable or unwilling to devote explicit effort to security issues. The latter two issues can be attributed to further differences to the laptop/desktop class of devices as discussed next.

- The user interface is – after battery runtime – the biggest remaining technical limitation of current mobile devices: touch screens with on-screen keyboards and/or small thumb keyboards are not sufficiently efficient for regularly entering long passwords. Even security conscious users will find it hard to justify the time overhead of entering passwords on these limited input methods. Security measures therefore have to cope with *user input limited in length and duration*.
- Mobile devices are by definition used in different locations and in different contexts. Therefore, users will rarely be able to commit their undivided attention to the use of their device, but will have to remain aware of their surroundings and focused on the real-life interactions (e.g. crossing a busy road while trying to authenticate to their smart phone to read the SMS that has just been received). For this reason, security measures need to be as unobtrusive as possible, or users will – if given the choice – simply deactivate them. Additionally, security relevant interaction will also take place in a mobile context, and eavesdropping-type attacks (referred to as shoulder surfing in the context of authentication methods) can therefore not be ruled out by assuming a secure location (such as the user's office or home). All security measures have to take the *multitude of potential contexts of use* into account.
- The proliferation of mobile devices gives a wider range of the world population access to informational services. Market analysis data suggests that we already have more smart phones accessing Internet services than all laptop/desktop type systems together. This implies that more end users with no training or previous exposure to computer systems use their mobile devices for security critical transactions. In the traditional domain of laptop/desktop systems, many users received some form of training on their first contact with computer systems, and these introductions often included a part on security best practices. With wide availability of cheap smart phones outside the traditional area of computer systems, the number of untrained end users grows significantly, and all security measures therefore have to become *intuitive*.

The implication of these differences from a security point of view might be summarized as "users don't care about security". We argue that this is not the case, but that products have not yet been able to provide a sufficient compromise between usability and security that takes into account all these differences from a usability point of view. Especially the two issues of untrained users being unaware of the implications of security measures and of limited attention to security measures are the main cause for the difficulty of establishing secure communication with end users.

V. Unsolved Problem: End-to-end Secure Channel to the User

With the technical approaches outlined above (section III), we can create a a chain of trust to secure a device platform

starting from its power-off state: under direct control of the legitimate owner and starting with secure hardware as trust anchor, a chain of signed code (boot loader, hypervisor, kernel, platform, apps) could prevent the installation and execution of malware. In combination with encrypted mass storage and secure key storage in an embedded smart card, a mobile device can implement a complete secure boot process and therefore provide a protected environment as long as all signed code is reasonably secure (apps do not have to be assumed universally secure).

Current approaches developed for the mass-market (e.g. ARM TrustZone and the Mobicore secure OS) provide a comparable technical solution and extend it with the secure user interface part executed alongside the main operating system so that the standard user interface elements (e.g. the Android UI elements) no longer have to be trusted (cf. Fig 1). However, two problems remain, as we have to assume malware running alongside trusted code in the form of untrusted third-party apps executed on a partially trusted (not malicious, but potentially exploitable) OS, and we have to assume users to be not as diligent as we would like them to be from a security point of view (section IV).

A. Securing output

Even with a secure UI (such as Mobicore) assisted by hardware virtualization (such as ARM TrustZone) and an embedded smart card (such as an NFC secure element), nothing prevents a malicious app from trying to fake the user interface that is normally presented by the secure UI and therefore manipulating output from a presumably secure app/service to the user. The reason is that both the trusted and untrusted software components rely on the same input/output modalities – the single touch screen in the case of current smart phones. Although some solutions have been developed towards a secure GUI on desktop-type systems (see e.g. the X11 windows extensions by Feske and Helmuth [15]), mobile devices require different approaches to visualization: window managers and the resulting window decorations are rarely available, and running applications often use full screen modes. Under the assumption of applications with access to all parts of the screen, we therefore have to deal with malware trying to copy the look and feel of other parts of the (secure) system.

We explicitly repeat that securing output to the user is significantly harder than communicating with a backend service or another device, because users cannot realistically verify digital signatures or perform decryption operations without the help of computing devices. Therefore, we cannot rely on the standard approach of using a cryptographic protocol to secure communication through untrusted components. Furthermore – considering the usability issues discussed above – end users would not devote the required effort and attention to verifying their end of a cryptographically secured communication even if they were capable of doing so.

The only possible solution to provide users with the required visual cue in an intuitive, unobtrusive, and low-effort manner seems to be additional hardware. Various options seem suitable, from a simple RGB LED that indicates which virtual guest is currently controlling the UI to a secondary display under exclusive control of the (limited) trusted components. However, the really interesting question is not the technical implementation, but standardization on the user interaction: will everybody agree on *one* standard that users can become used to and that they will actually check (unobtrusive, but noticeable for every secure interaction, intuitive, understandable, and documented for first-time users)?

B. Securing input

We face an equivalent problem for input from the user to a presumably secure app/service: all input is currently handled by the main OS because of its exclusive hold on the single touch screen. Secure input needs a way for the user to a) be sure that input only goes to secure components (e.g. trusted virtual guest domains) and b) to initiate a switch between different components (virtual guests) that is not subject to an app-in-the-middle attack.

While this can also be achieved with additional hardware such as dedicated pinpads hard-wired to the SE, practical experience strongly suggests that users will not want their mobile devices to become bigger "just for security purposes". A more practical approach is therefore to first secure output and then rely on the intuitive end-user assumption that those components that produce the current output will also receive the input. This solves the problem concerning the user interaction aspect by giving (secure) feedback to users which component (virtual guest) they are interacting with. However, the technical implementation still remains open for current smart phone platforms, because the touch screen driver will typically reside in the main OS and not in the limited hypervisor code. Our current approach is to move touch screen event handling into the hypervisor code and forward those input events only to the virtual guest that is currently controlling screen output. Implementation details and their implications are subject to future research.

VI. Conclusions and Outlook

After systematically identifying threats to user data on current mobile devices, we have analyzed the landscape of technical approaches to addressing them and the significant differences to traditional security measures from a security point of view. To this end, we found the need for additional technical components, namely virtualization, further integration of existing embedded smart cards with secure boot verification, and secure user interface components. The biggest open issue is how to design the user interaction in

terms of data input and output in such a way that users can reliably and unobtrusively be aware of which application part they are communicating with. This is currently unsolved and remains the most important research question to advance the area of secure mobile devices.

In a cooperation with other research groups, we are currently preparing an extensive user study concerning the granularity of separating different parts of mobile platforms/apps and on visualization and input concepts. Results will be published as future work and we expect them to inform our architecture decisions towards addressing the question of intuitive secure user interaction.

ACKNOWLEDGMENTS

This work has been carried out within the scope of *u'smile*, the Josef Ressel Centre for User-Friendly Secure Mobile Environments. We gratefully acknowledge support by the Christian Doppler Gesellschaft, A1 Telekom Austria AG, Drei-Banken-EDV GmbH, LG Nexera Business Solutions AG, and NXP Semiconductors Austria GmbH.

REFERENCES

[1] *Community framework for electronic signatures*, OJ L 13 of 19.1.2000, European Parliament and European Council Directive 1999/93/EC, January 2000. [Online]. Available: http://europa.eu/legislation_summaries/information_society/other_policies/l24118_en.htm

[2] A. Egners, U. Meyer, and B. Marschollek, "Messing with Android's permission model," in *Proc. TrustCom 2012*. IEEE CS Press, 2012, pp. 505–514. [Online]. Available: http://dx.doi.org/10.1109/TrustCom.2012.203

[3] S. Höbarth and R. Mayrhofer, "A framework for on-device privilege escalation exploit execution on Android," in *Proc. IWSSI/SPMU 2011, colocated with Pervasive 2011*, June 2011.

[4] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2009. [Online]. Available: http://www.bitcoin.org/bitcoin.pdf

[5] A. Egners, B. Marschollek, and U. Meyer, "Hackers in your pocket: A survey of smartphone security across platforms," RWTH Aachen, Tech. Rep. AIB-2012-07, May 2012. [Online]. Available: http://itsec.rwth-aachen.de/publications/ae_hacker_in_your_pocket.pdf

[6] M. L. Polla, F. Martinelli, and D. Sgandurra, "A survey on security for mobile devices," *IEEE Communications Surveys & Tutorials*, vol. 15, pp. 446–471, February 2013.

[7] A. K. Karlson, A. B. Brush, and S. Schechter, "Can I borrow your phone? understanding concerns when sharing mobile phones," in *Proc. CHI 2009*. ACM Press, April 2009, pp. 1647–1650.

[8] M. Roland, J. Langer, and J. Scharinger, "Practical attack scenarios on secure element-enabled mobile devices," in *Near Field Communication (NFC) Workshop 2012*, March 2012, pp. 19–24.

[9] C. Rudolph, "Covert identity information in direct anonymous attestation (DAA)," in *Proc. IFIP SEC 2007*, ser. IFIP, vol. 232. Springer-Verlag, May 2007, pp. 443–448.

[10] J. Brakensiek, A. Dröge, M. Botteck, H. Härtig, and A. Lackorzynski, "Virtualization as an enabler for security in mobile devices," in *Proc. IIES 2008*. ACM Press, 2008, pp. 17–22. [Online]. Available: http://doi.acm.org/10.1145/1435458.1435462

[11] J. Seifert, A. D. L. B. Conradi, and H. Hussmann, "TreasurePhone: Context-sensitive user data protection on mobile phones," in *Proc. Pervasive 2010*, ser. LNCS, vol. 6030. Springer-Verlag, 2010, pp. 130–137.

[12] R. Mayrhofer, J. Fuss, and I. Ion, "UACAP: A unified auxiliary channel authentication protocol," *IEEE Transactions on Mobile Computing*, vol. 12, no. 4, pp. 710–721, April 2013.

[13] R. Findling and R. Mayrhofer, "Towards face unlock: On the difficulty of reliably detecting faces on mobile phones," in *Proc. MoMM 2012*. ACM Press, December 2012.

[14] O. Riva, C. Qin, K. Strauss, and D. Lymberopoulos, "Progressive authentication: deciding when to authenticate on mobile phones," in *Proc. USENIX 2012*. USENIX, August 2012.

[15] N. Feske and C. Helmuth, "A Nitpicker's guide to a minimal-complexity secure GUI," in *21st Annual Computer Security Applications Conference (ACSAC)*, 2005.