J Supercomput (2011) 55: 207–227
DOI 10.1007/s11227-010-0407-1

An approach to introducing locality in remote attestation using near field communications

Ronald Toegl · Michael Hutter

Published online: 19 March 2010

Abstract Remote Attestation, as devised by the Trusted Computing Group, is based on a secure hardware component—the Trusted Platform Module (TPM). It allows to reach trust decisions between different network hosts. However, attestation cannot be applied in an important field of application—the identification of physically encountered, public computer platforms. Unfortunately, such computer terminals are especially exposed and the software running on them cannot be assumed unaltered and secure.

Three challenges arise. The cryptographic protocols that actually perform the attestation do not provide for human-intelligible trust status analysis, easily graspable conveyance of results, nor the intuitive identification of the computer platform involved. Therefore, the user needs a small portable device, a token, to interact with local computer platforms. It can perform an attestation protocol, report the result to the user, even if the display the user faces cannot be trusted and may be connected to the platform under scrutiny. In addition, the token must establish that the particular machine faced actually contains the TPM that performs the attestation.

In this paper, we demonstrate an attestation token architecture which is based on a commodity smart phone and which is more efficient and flexible than previous proposals. Furthermore, we introduce an autonomic and low-cost Near Field Communication (NFC) compatible interface to the TPM that provides a direct channel for proof of the TPM's identity and local proximity to the attestation token.

Keywords Trusted computing · Remote attestation · Near field communication

R. Toegl (✉) · M. Hutter
Institute for Applied Information Processing and Communications (IAIK), Graz University of Technology, Inffeldgasse 16a, 8010 Graz, Austria
e-mail: ronald.toegl@iaik.tugraz.at

M. Hutter
e-mail: michael.hutter@iaik.tugraz.at

1 Introduction

Security enforced by software can be manipulated by software-based attacks. To overcome this dilemma, the Trusted Computing Group (TCG) has defined a set of specifications of which the *Trusted Platform Module* (TPM) is the central component. It is a system component deeply embedded in a machine's hardware and software architecture. One of its mechanisms, called *Remote Attestation*, reports the platform's state to another host on the Internet. This helps to establish cryptographically qualified and tamper-evident assurance about the software configuration of a machine.

As attestation allows to determine the absence of malicious software, it is desirable for a user to perform it prior to providing sensitive or confidential data to a computer system. This is especially interesting for computers openly available in public places where users simply walk up to such machines when they need the services offered. Currently, such computers are highly exposed and threatened by a variety of software-based attacks in the form of viruses, keyloggers and root kits. Therefore, public systems for ad hoc use cannot be trusted to handle sensitive information such as account logins, passwords or other private data—unless their configuration were properly attested and found secure for the user's purposes. However, several additional challenges need to be overcome to achieve a telling indication of trustworthiness in a usage scenario where the user physically confronts the system.

While the TPM may be trusted to perform securely as specified, it does not offer a secure local interface to display the gathered results to a user. Therefore, the need for a trusted display or an indicative token arises, lest a malicious public machine fakes reports on its state. As McCune et al. [26] point out, it would be desirable to equip the user with an ideal, axiomatically trustworthy device, which they call *iTurtle*. It would then indicate the security of a device to the user. A more practical implementation of attestation in kiosk scenarios using off-the-shelf smart phones is demonstrated by Garriss et al. [14]. Still, the TCG's attestation protocol does not guarantee that the TPM is located within the machine the user faces. Following this insight, Parno [30] proposes a direct link between the user and the TPM, so that human interactors can themselves establish the proximity of the attesting machine.

In this article we build on these previous results and introduce two novel improvements. First, considering the resource limitations of mobile devices, currently proposed schemes are not flexible and scalable enough. We demonstrate an efficient, user-friendly solution based on smart phones and a trusted third party. Second, to include a proof-of-locality in the process, we propose to introduce Near Field Communication (NFC) technology in the TCG's security architecture and present a proof-of-concept implementation.

The remainder of this article is organized as follows. In Sect. 2, we outline Trusted Computing and NFC technology as well as Remote Attestation. We discuss related work and present the locality-aware scenario we consider, followed by an introduction to our approach, in Sect. 3. In Sect. 4, our Mobile Attestation Token Architecture is presented. In Sect. 5, we discuss the integration of NFC in the TPM, introduce new TPM commands, and present the definition of an air-interface protocol between the TPM and a mobile NFC device. Section 6 gives implementation details of the Mobile Attestation Token and our NFC demonstrator. The paper concludes in Sect. 7.

2 Preliminaries

2.1 Trusted platform module

Trusted Computing as it is available today is based on specifications of the *Trusted Computing Group*[1] *(TCG)*. A hardware component, the *Trusted Platform Module (TPM)* [36], is integrated into commonly available general purpose hardware, with hundreds of millions of platforms shipped so far. Similarly to a smart card, the TPM features cryptographic primitives but is physically bound to its host device. A tamper-resilient integrated circuit contains implementations of public-key cryptography, key generation, cryptographic hashing, and random-number generation. With these components the TPM is able to enforce security policies on hierarchies of secret keys to protect them from software attacks by any remote attacker. One such key, the Endorsement Key, provides the TPM with a unique identity.

The TPM not only holds cryptographic key material, but also guarantees the integrity of measurements of the host system's state. This state information is held in a few specially protected *Platform Configuration Registers (PCRs)*, which can only be written via the one-way *extend* operation. A PCR with index i in state t is extended with input x by setting

$$PCR_i^{t+1} = \text{SHA-1}(PCR_i^t || x).$$

The PCRs can be used to exactly describe the software executed on a machine by following the transitive trust model, where each software component is responsible to measure other components before it invokes them. In the simplest case, the caller computes a chain of hash values and extends a PCR with the result, starting from the BIOS, covering bootloader, kernel, system libraries, etc., up to application code, before any of this executable code is allowed to run. A variant which allows *dynamic* code measurements at runtime is available on systems with hardwired chipset-CPU-TPM cooperation.

Ultimately, a *chain of trust* is established where the exact configuration of the platform is mapped to PCR values. If such a system state fulfills the given security or policy requirements, we refer to the system state as a *trusted state*.

Another high-level feature of the TPM is that it can *bind* data to a platform by encrypting it with a *non-migratable* key. Such a key never leaves the TPM's protected storage unencrypted. An extension to this is *Sealing*. A key may be sealed to a specific (trusted) value of the PCRs. A sealed key will not leave the TPM and is only used by the TPM if the PCRs hold the same values which were specified when the key was sealed. Thus, use of the key can be restricted to a trusted state of the computer.

The hardware resources of a TPM are manufacturer-implementation specific and typically very limited. For instance, the TPM supplies only a few cryptographic key slots and continually swaps keys to and from external storage during operation. To document the creation of the hash chain and for later analysis of the aggregated PCR information, a Stored Measurement Log (SML) needs to be kept on the host system.

[1] http://www.trustedcomputinggroup.org/.

The current TPM design establishes the need for a singleton system software component to manage the TPM device resources and arbitrate concurrent accesses. To this end, the TCG specifies an architecture that implements TPM access and management, the *TCG Software Stack* (TSS) [35], which covers operating system and applications' support.

2.2 Platform support for remote attestation

The TCG standards describe a set of compact basic hardware building blocks that are designed to enable a host to measure the exact software binaries running on it and also to generate an authentic report on this. Prominently, in the `TPM_Quote` operation, the TPM signs the PCR status. On the Internet, this enables the more complex protocol of *Remote Attestation*.

Here, a *Remote Verifier* may challenge a TPM equipped platform to expose its system state. The challenge includes a fresh nonce, which is passed on to the TPM for calculating the Quote result. The TPM signs a set of current PCR values and the nonce with a hardware-protected key. The result is then returned to the verifier together with the SML and a set of credentials, which describe the platform hardware and the TPM's implementation security and specification conformity. The verifier can now analyze the state of a platform and assess the trustworthiness of its configuration for a given purpose.

The TCG remote attestation architecture requires that the host sends a very detailed description of its system state to a verifier. In general, this would raise the question of privacy protection, if the unique identity of the TPM were to be used. Therefore the Endorsement Key, injected by the TPM manufacturer, or created upon machine deployment, is not used for this signature. Rather, only pseudonymous keys may be applied. Such an *Attestation Identity Key (AIK)* can be created on demand within the TPM. Its authenticity can then be certified by an online trusted third party, called PrivacyCA, demonstrated in [31] or by the more complex group-signature based DAA scheme [3]. An AIK can be used one or several times, whenever an attester requests the attestation of the trusted platform. Verifiers can determine the correctness of the signature after confirming the validity of the certificate and querying a revocation service.

Essential to the overall attestation process is to collect and later analyze a complete and meaningful set of state information. As of recently, hardware manufacturers provide BIOS support to measure the first phases of system boot into the TPM. When control is passed over to the bootloader, it also takes over the responsibility to measure the next component, i.e. the operating system kernel. Trusted Grub,[2] the Open Secure LOader[3] (OSLO) and Trusted Boot[4] (tboot) demonstrate this. Operating systems that measure at least a partial chain-of-trust have been demonstrated, e.g. in [33]. Still, using only the quote result and measurement log, coming to a trust decision remains a tedious and complex task and the number of possible combinations of

[2] http://sourceforge.net/projects/trustedgrub/.

[3] http://os.inf.tu-dresden.de/~kauer/oslo/.

[4] http://sourceforge.net/projects/tboot/.

secure software configurations in today's open system architectures is often too large in practice [12].

A technology that helps to reduce the length and complexity of the chain-of-trust and therefore eases analysis of system states is virtualization. Virtualization is a methodology of dividing the resources of a computer into multiple execution environments, by applying concepts such as hardware and software partitioning, time-sharing, machine simulation or emulation. A single *hypervisor* runs directly on the hardware, which it manages exclusively. It then provides for the creation, execution and hibernation of isolated *compartments*, each containing an unmodified operating system.

Recently, hardware platforms [7, 8] have become available that support the security of virtualization by providing strong isolation of compartments on commodity hardware. Those platforms also extend the basic TCG model of a *static* chain-of-trust from hardware reboot onwards. Instead, they provide the option of a *dynamic* switch to a trusted system state. A special CPU instruction allows to switch the system into a well-defined secure state and then measure and run a piece of software, typically a hypervisor, which has full system control. Close hard-wired cooperation of CPU, chipset and TPM guarantees that the result is accurate. Thus, influences from boot-time only components as for example the BIOS can be prevented. Another challenge is keeping track of known good system PCR configurations. This proves extremely challenging due to the high number of possible combinations found in today's complex and ever-updated operating systems [12]. As a workaround, with virtualization in place the hypervisor can perform a single measurement of the complete compartment file instead. This greatly simplifies the comparison against known good values and thus allows practical application of mechanisms like Sealing and Remote Attestation. A further advantage of combining virtualization with TC it that it allows separating the execution environments of different applications. This naturally results in a higher attack resistance and availability, as it allows to contain software attacks to a single compartment. Demonstrations of such systems are Terra [13], or the results of [6], or the EMSCB [11] and OpenTC [28] projects.

Alternative concepts to reach meaningful conclusions at state analysis include *Property-based Attestation* [5, 21, 32]. Here the state analysis is delegated to a specialized *Trusted Third Party* (TTP) which issues certificates for specific properties.

Note that the TC concept does not claim to enforce perfect security under all conditions and tasks but defines a trustworthy system as *a system that behaves in the expected manner for the intended purpose*. Furthermore, if a remote machine is under full physical control of the attacker, a simple hardware TPM reset attack [20] would allow feeding fake measurements to the TPM, circumventing the chain-of-trust altogether.

2.3 Near field communication

Near Field Communication (NFC) is a wireless communication technology that provides a platform for many applications such as mobile ticketing, contactless payment, and interactive smart posters. One key feature of NFC is the simple data acquisition just by touching an object with an NFC-enabled reader. Such readers might be integrated in mobile phones or digital cameras that transfer information to the devices

in their proximity. There exist two different modes for a communication between a reader (initiator) and a target device. In passive communication mode, the initiator provides an electromagnetic (EM) field which is used to power the target device and which allows a bidirectional communication. In active communication mode, both the initiator and the target device provide an alternately generated EM field so that both devices require an active power supply.

NFC is based on the Radio Frequency Identification (RFID) technology that operates at 13.56 MHz frequency. As opposed to RFID, NFC follows several specifications that have been standardized by the International Organization for Standardization (ISO) and the European Computer Manufacturers Association (ECMA). These standards specify the used data modulation, coding, frame formats, data rates, and also the application-specific transport protocol. The NFC interface and protocol (NFCIP-1) is standardized in ISO/IEC 18092 and ECMA 340 and also in ISO/IEC 21481 and ECMA 352 (NFCIP-2). In addition to these specifications, there exist some definitions from the NFC Forum which is a non-profit organization that promotes the use of NFC in electronic devices. Among many other definitions, they defined a common frame format for transmitting different media types such as Multipurpose Internet Mail Extension (MIME) objects and Uniform Resource Locators (URL). This frame format is called NFC Data Exchange Format (NDEF) and can be used to automatically start an application on a mobile phone or to display a message after touching a target object.

In contrast to other wireless communication technologies which are designed for a large communication range, NFC enables short-distance communication between electronic devices. The typical operating distance between two NFC devices is only a few centimeters (up to 10 cm). Thus, a fixed location of an NFC tag (passively or actively powered) can provide evidence whether a mobile NFC device (or its user) has been at that location. Besides this evidence, NFC offers a very intuitive way for the user to communicate with a target object by simply bringing the devices close together (touching). It follows the very natural principle for communication between only two, locally present entities.

NFC offers a wide range of new applications with a key focus on easy-to-use products and touch-based solutions. Since 2004, NXP Semiconductors, Nokia, and Sony invited many other global leading companies from mobile industry, electronics, payment services, and multimedia enterprises to join the NFC Forum which already has more than 150 members from around the world.

3 Attestation of local platforms

Remote Attestation, as devised by the TCG industry consortium, achieves trust decisions between different network hosts. However, it cannot be applied in an important field of application—the identification of physically encountered computer platforms and their security status to the human user. The cryptographic protocols that actually perform the attestation do not provide for human-intelligible trust status analysis, easily graspable conveyance of results nor the intuitive identification of the computer platform involved. Therefore, the user needs a small portable device, a token, to interact with local computer platforms. It can perform an attestation protocol, report

the result to the user, even if the display the user faces cannot be trusted and may be connected to the platform under test.

3.1 Related work

In recognition of this, [26] propose the concept of an *iTurtle* device which should by design be trusted axiomatically. To achieve *user-observable verification* it should be as simple as possible, even without support for cryptography and thus easy to understand and certify. The authors envision a USB device with a mere two red and green LEDs indicating the trust status. The authors argue that integration in the TCG's cryptographic scheme would be too complex and that the challenge of state analysis on a restricted device would remain. In a more practical approaches, powerful PDAs and smart phones have demonstrated [29, 34] their applicability as trusted portable devices to work in conjunction with a trusted server and an untrusted public terminal to act as a secure keyboard and GUI to the user.

In a first combination with the TCG architecture, [25] demonstrate a mobile phone application which uses 2-D barcodes on stickers to identify a public key of devices like printers or IEEE 802.11 access points. The authors also consider integration in TPM-based attestation protocols. This early architecture does not guarantee the identity and standard-conformance of the TPM, i.e. lacks the proper use of the AIK credentials.

The specific case of attesting a public kiosk computer as available in lobbies or transportation terminals has been studied in detail by Garriss et al. [14], also using a mobile phone. A user wishing to use a kiosk first uses the camera of her smart phone to scan the barcode containing the hash of the AIK certificate of the kiosk. The phone then connects to the kiosk using Bluetooth. The kiosk now transmits the set of configurations it supports. The set is predefined and signed by the kiosk's operator, which has to be trusted. Now the user chooses a configuration and the kiosk reboots to build a fresh chain-of-trust. After it is online again, the phone performs an attestation protocol, compares the reported configuration against the chosen one and validates that the `TPM_Quote` result is indeed signed with the same AIK. The user is informed of the result, i.e. the trust status is displayed on her phone. She can then use the kiosk's applications or even take advantage of the virtualized kiosk software architecture. Here, the user may supply a private virtual machine image containing her choice of software and data, cf. [4]. In the end, the user logs out.

An alternative approach to the kiosk scenario is presented by Bangerter et al. [2]. They coin the term "ad hoc attestation" for their scheme which is based on a commercially available token equipped with an optical sensor and a display. It generates a nonce which is manually entered into the kiosk by the user. Due to the restricted hardware resources of the token, the state analysis is performed on a special server, which then *binds* the trust decision result to the kiosk's TPM. To complete the protocol, the user needs to point the token's sensor to the kiosk display. The kiosk then unbinds and communicates the trust result via flickering black-and-white images to the token. Thus, the server is able to remotely trigger the token, to display the cryptographically protected result. Note that the user is restricted in his choice of trustworthy configurations, as he has to trust in the commercial token server operator who performs

the decisions in his stead. Due to the combination of the attestation protocol with the binding mechanism, this scheme achieves protection against "*platform-in-the-middle attacks*."

Those attacks have been analyzed by Parno [30], who calls them *cuckoo attacks*. These schemes are sometimes also known as Mafia fraud attack or chess grandmaster problem. In essence, an attacker uses an honest entity as oracle to solve the challenges of her target. In practice, malware on a compromised local machine relays TPM messages to another TPM on a remote machine which is in a trusted state. The author concludes that a local binding between user and TPM is needed. If the user is in possession of a trusted hand-held device, this may be achieved physically via a special hard-wired interface or cryptographically by providing users with a key by means of a sticker on the machine casing. Li et al. [22] describe adding a serial wired interface to the TPM to access protected data for backup and also for providing authorization to a few restricted operations that require the physical presence of the TPM owner. No additional functionality is offered to users.

3.2 Open challenges

Based on the presented literature we identify the following additional challenges for the ad hoc attestation of physically present computers, which occur due to the locality of the attestant.

- *Flexible and Scalable Trust Decisions.* The display of a public computer must not be trusted to securely show the trust decision. To convey the trust status to a user, a mobile attestation token is needed that provides a suitable display and a secure communication channel to the TPM. Several existing implementations not only display the result but also perform the trust decision on the mobile attestation device. The performance of these devices limits the size of the known-good-value repository and the complexity of the state analysis needed for the trust decision. Also, if only a small set of possible configurations is provided, none of them might match the specific security requirements of the user. A priori stored reference values also limit the flexibility in case of system updates, or when encountering terminals from unexpected operators. The same holds true for proprietary servers, which force a user to trust in their operator and his policies.
- *Direct, Local Channel between User Token and TPM.* Practical proposals have so far considered different interfaces such as Bluetooth or USB. However, as outlined in [30], both technologies require an honest software stack to forward their messages to the passive TPM device. A direct wired physical channel would require extensive changes to the TPM design and new standard plugs, both being expensive and impractical. Flickering displays provide only one-way communications and require users to obtain a functionally-closed, special-purpose token. Bluetooth has a long radio range and thus it could also connect to a neighboring kiosk. To prevent this, current proposals introduce stickers that identify TPM keys. However, stickers are easy to manipulate [23]. Foremost, it is extremely easy to copy them (with the attacker posing as a legitimate user, taking the photo with his mobile phone camera) and thus fake the identity of another kiosk. This is exactly the setting for the cuckoo attack we need to prevent.

3.3 Scenario

Attestation is useful to improve the security for a number of computing services, including not only remote but, as we believe, also physically present systems. In general, various types of systems may be encountered in different usage scenarios.

For instance, a user might want to learn if a public general-purpose desktop computer is secure for ad hoc use. Customers would like to be assured that a point-of-sales terminal in a shop will not collect their PIN together with the information on the magnetic stripe of their credit card for later frauds. The same holds true for other types of Automatic Teller Machines (ATMs) and payment terminals. Vending machines could be reconfigured by attackers to collect cash but not to release their goods; local attestation could prevent this, too. Other security critical applications may also be found in embedded systems or even peripherals like printers or access points. Here, a service technician might find a method to identify the exact software configuration and its integrity to be useful. Giving voters a method to validate that electronic voting machines have not been tampered with might assist to add trust to a poll's outcome.

For easier illustration of our approach in the remainder of this article, we will limit our description to just one specific and instructive scenario, kiosk computers. Our solution does not depend on assumptions defined for confidential architectures as found in banking applications or strict legal requirements as required for eVoting solutions. We believe that the proposed approach can be modified to specific needs of such other, more specialized, scenarios with reasonable effort.

3.3.1 Kiosk computing

Kiosk computers are often found at shops, in hotel lobbies, transportation terminals or Internet cafés; they are public terminals to provide applications like web browsers or ticket-vending services. Such kiosks are rarely deployed alone; in most cases several similar devices are operated in close vicinity. We assume that the kiosk is equipped with persistent storage, e.g. a hard drive, and can therefore store data and programs over a power cycle. A result from the public availability is that also attackers can visit the kiosk repeated times during operation hours and pretend to be legitimate users. Attackers are assumed to have full control over the software running on the kiosk, thus software cannot be trusted at all and keyloggers and fake security tools must be assumed. We further assume that wireless communications can be eavesdropped.

With TPM-based attestation we desire to provide the user with means to establish trust in such a device, but of course we have to consider the limitations of the TCG's architecture: it is not designed to protect against hardware attacks. We assume that this is compensated by operational measures, i.e. even if the devices may be unattended, they will be physically protected, i.e. by robust casings fixed to the ground, integrated keyboards and displays that prevent hardware-based attacks. Also, we assume that the operator performs hardware and software maintenance on a regular basis, thus making most hardware attack schemes, like adding malicious devices to the casing, impractical.

Figure 1 further illustrates the scenario we envision. It shows the following four different situations.

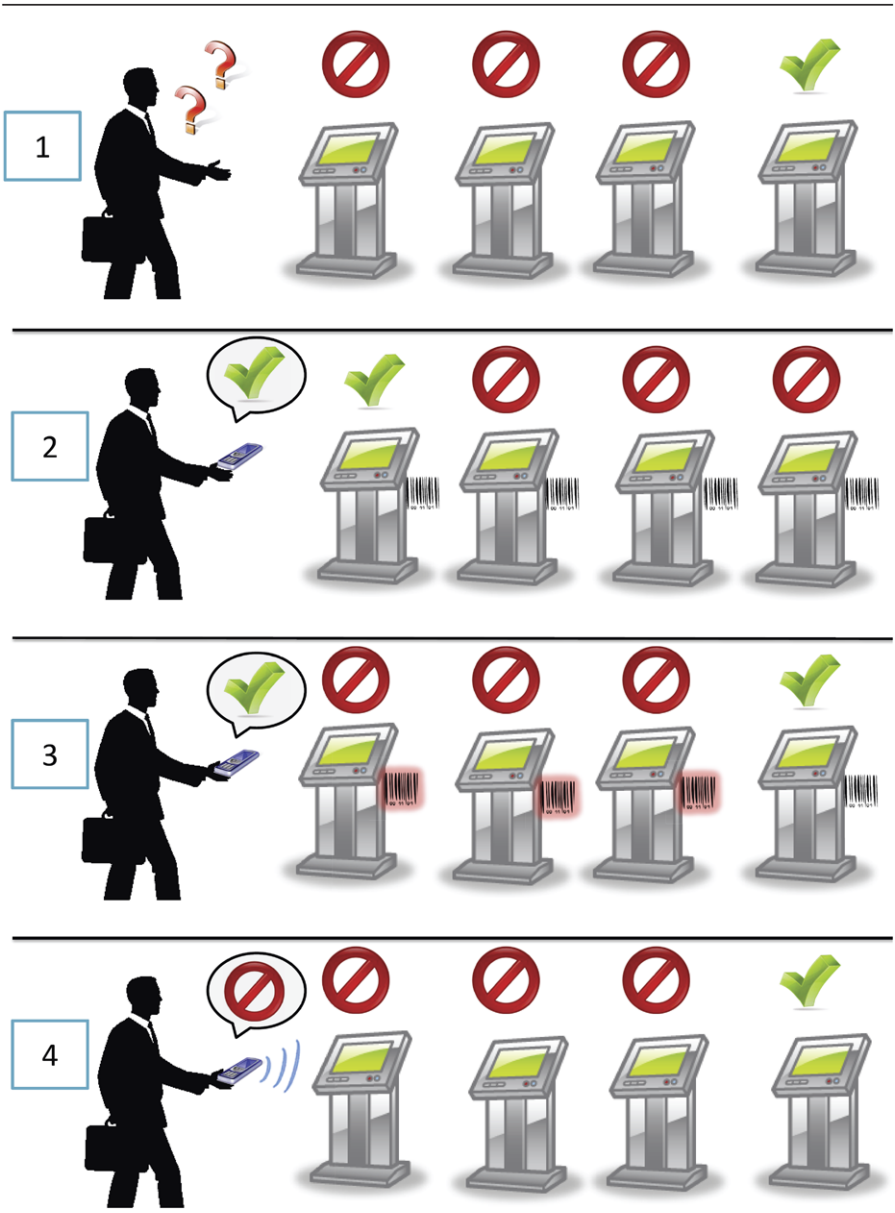

Fig. 1 Description of the motivating user-case (*1*), proposed solutions from the literature (*2*), a possible attack scenario (*3*), and our proposed solution (*4*)

1. When a user encounters a room with several terminals available, he has no means to establish the trustworthiness of a single device. Therefore he might choose a malicious kiosk computer for his task, unmeaningly exposing private information to an attacker.

2. Here, the user is equipped with a smart phone which serves as attestation token. We illustrate a successful attestation similar to when carried out with the scheme that has been demonstrated by [14, 25]. Note that the individual kiosks are identified by barcode tags attached to them.
3. In this situation the unprotected kiosk identification tag has been manipulated, respectively copied, by an attacker in such a way that the user believes to communicate with the unmodified kiosk, while he in fact faces a malicious machine. Due to the long range of Bluetooth communications, this attack compromises the scheme of [14].
4. The last situation illustrates our proposal, where NFC is used to establish which is the kiosk that actually performs the attestation. With this mechanism in place, the user can recognize that the first machine is not secure and will refrain from performing any critical task on it.

We now present two novel improvements which constitute our main contributions. In the next section, we outline a kiosk attestation architecture which is designed to be user-controlled, flexible and scalable with regard to kiosk state analysis. In Sect. 5, we will detail how NFC can be integrated in attestation, thus providing for direct user token to TPM communications.

4 Mobile attestation token

4.1 The MAT protocol

In our scheme, three parties collaborate to perform a cryptographic protocol. The *Kiosk* contains a TPM and an operating system that offers a complete chain-of-trust and measurement services that allow the extraction of properties. Secondly, the *Mobile Attestation Token* (MAT) is the client the user installs on his mobile phone. Finally, we introduce a trusted third party, the *Verification Server* (VS). The protocol flow is shown in detail in Fig. 2.

0. In an initial setup step, the user chooses the VS she trusts and configures it with policies according to her preferences and needs. She then transfers the public-key certificates and the URL of the VS to the MAT.

1. When at some later point in time encountering a public computer, the user connects the MAT to the attestation service installed on the kiosk.
2. In the beginning of the attestation protocol, the MAT generates a nonce N_a to provide fresh data for replay protection.
3. With the first message to the kiosk, the user initiates the attestation with her MAT, transmitting the *URL* of the Verifications Server she intends to use and N_a.
4. The kiosk will then gather a quote of its recorded system state from the TPM.
5. This quote, i.e. $Sig_{AIK}(PCR, N_a)$, the signature over all PCR registers under an Attestation Identity Key *AIK* of the TPM, is then returned to the kiosk.
6. Now, the kiosk establishes a secure TLS connection to the VS via the *URL* provided by the MAT. It transmits the quote together with the AIK certificate and the stored measurement log *SML* which documents the chain-of-trust of the kiosk in detail.

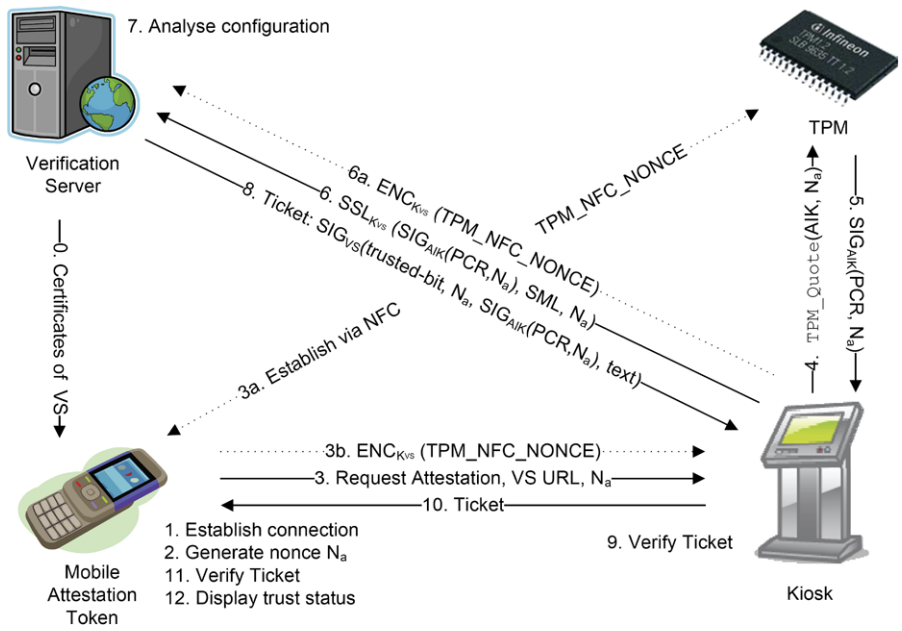

Fig. 2 Ticket-based Local Attestation scheme with MAT and trusted Verification Server. The optional steps *3a*, *3b*, and *6a* are only performed with the NFC extension as described in Sect. 5.3

7. With complete information, the Verification Server analyzes the quote and decides the trustworthiness according to the detailed requirements of the user, using its local or other available Known-Good-Value services or property extraction modules. It also validates that *AIK* is indeed a TPM protected key and asserts that the PrivacyCA did not revoke the certificate.
8. Once a trust decision is made, the VS assembles and returns a ticket to the kiosk: $SIG_{VS}(trusted, N_a, Sig_{AIK}(PCR, N_a), text)$. It contains a binary trust decisions (the `trusted`-bit) and a free `text` for additional messages.
9. The ticket is validated on the VS before being passed on to the MAT. This allows a kiosk operator to gather statistics on how much acceptance the offered configuration finds.
10. The unmodified ticket is passed on to the MAT.
11. The MAT verifies the signature using the pre-installed VS certificate and also that N_a is from the same session.
12. If this succeeds, it can then proceed to finally display the result to the user. Here, intelligible icons can be used to illustrate the `trusted`-bit.

4.2 Discussion

The protocol attests the kiosk configuration to the Mobile Attestation Token. The central design decision is to delegate the complex state analysis and decision procedure to the Verification Server. The capability to decide about the trustworthiness of the

kiosk is not limited by the restricted resources on the mobile device. Therefore the number of kiosk configurations (i.e. hash-based measurements, properties extracted, validation of signed code, etc.) considered will not influence the local performance of the protocol. The online VS will also be able to check the validity of AIK certificates online, so that kiosk identities need not be specified by the user beforehand. Indeed, infinitely many attestation identities may be used, if a trusted PrivacyCA vouches for them. With all this performed remotely, our scheme requires the MAT only to validate the signature of the ticket, the nonce and the trusted bit. The protocol therefore *scales* with the number of kiosks and their configurations.

Also our protocol does not require any vendor-specific operations or tokens—it can be implemented on any device capable of the small set of cryptographic functions and the necessary communication interfaces. In addition, there are no limitations on who operates the Verification Server. It could be the kiosk operator as well as the user or any commercial or open institution. The VS might even consult other services to help him decide on a reported system state. This *flexibility* not only allows for sound trust decisions, but also provides easy adaption to changing profiles.

Thus, our proposal is not only a scalable and flexible technical solution, but at the same time it also protects the users' right of self-determination in their trust decisions.

5 An NFC interface for the TPM

In this section, we outline how the TPM could be extended with an NFC interface to create a direct channel to the MAT. We believe that this will not require extensive changes to the TPM design. NFC has been designed to be integrated in small hardware solutions like smart cards which are very similar to many TPM implementations. Furthermore, many of the challenges that have to be overcome in the design of passive NFC tags are not an issue with the TPM. For instance, the TPM has an active power supply and full cryptographic capabilities. Only a simple, passive RF interface is needed, and the antenna circuit could just be printed on the mainboard of the host machine.[5] We believe that an NFC interface is cheaper than a proprietary wired interface, which would require modifications to TPM, board, casing and MAT.

In this way it is possible to establish a direct link from the attestation device to the TPM. Note that in our approach any software on the kiosk is circumvented, making any kind of software-based attacks on the connection impossible.

5.1 New TPM commands

Also changes to the TPM itself can be limited to a minimum. As a special-purpose trusted component, it should not provide more features than necessary to perform its tasks and therefore should not operate as a full flexible NFC reader to the host. Also, changes to the TPM API should be minimal and not affect normal operations. For brevity, we only present the changes to the current TCG TPM specification [36] in this section.

[5] Assuming a non-shielded casing.

Table 1 The TPM_establishNonce_NFC command establishes a shared nonce between remote NFC reader and TPM. The resulting nonce is not returned to the host machine but retained in the protection of the TPM

Incoming				
Parameter	Size	Type	Name	Description
1	2	TPM_TAG	tag	TPM_TAG_RQU_COMMAND
2	4	UINT32	paramSize	Total number of input bytes including paramSize and tag
3	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_establishNonce_NFC
Outgoing				
Parameter	Size	Type	Name	Description
1	2	TPM_TAG	tag	TPM_TAG_RSP_COMMAND
2	4	UINT32	paramSize	Total number of input bytes including paramSize and tag
3	4	TPM_RESULT	return code	The return code of the operation
4	4	TPM_COMMAND_CODE	ordinal	Command ordinal: TPM_ORD_establishNonce_NFC

The RF interface is to be activated only in Enabled-Activated-Owned state of the TPM life cycle and an owner-authorized call to TPM_SetCapability is needed to activate the permanent flag enableNFCInterface that enables the following operations.

We introduce a new command that allows the NFC reader and the TPM, which have no prior knowledge of each other's identity, nor a shared key, to jointly establish a shared secret over the NFC channel. This secret can then be used as nonce in a *single* subsequent TPM operation. The TPM_establishNonce_NFC command is described in Table 1. It is important to notice that if the command returns with TPM_SUCCESS, the nonce is not returned to the TPM's host machine but retained in a special volatile and protected register TPM_NFC_NONCE inside the TPM. This register can be read-accessed as if it was an additional PCR, but with one exception: it is always reset to zero after a read operation. If the protocol fails, or times out, appropriate error codes are returned. TPM_establishNonce_NFC does not require authorization, as it only stores the nonce. All commands that use its result must be properly authenticated. The command itself performs a standard Diffie–Hellman key-exchange operation which is described in detail in the next section.

Minor changes are now needed for TPM commands that utilize this nonce, for instance, TPM_Quote. It is called with a TPM_PCR_SELECTION that indicates the PCRs to consider. The behavior is extended as follows: TPM_NFC_NONCE is selected like other PCRs with index: number of normal PCRs + 1. If TPM_NFC_NONCE is zero, the command terminates with error code TPM_NO_NFC_NONCE, else its value is hashed together with the PCRs and signed with the provided AIK. The values of all used registers are returned to the host. The TPM_NFC_NONCE register is then set to zero.

This way, the quote result depends on the `TPM_NFC_NONCE` that was previously agreed upon by the NFC reader, i.e. the Mobile Attestation Token and the TPM. As the quote result is signed with an AIK, this links `TPM_NFC_NONCE` to an authentic TPM. Each nonce can only be used once, thus guaranteeing freshness. Other commands which access PCRs can be adapted in a similar way, without changing their signature.

5.2 The NFC nonce-agreement protocol

Essentially, the security of NFC is based on the physical characteristics of the electromagnetic near-field, which limit the operational range to about 10 cm. Still, eavesdropping attacks remain a threat in NFC applications that can be performed even at a distance [15].

In order to establish a shared nonce between the TPM and the NFC-enabled reader, we propose to use a classical key-agreement scheme according to Diffie and Hellman (DH). This scheme is quite simple to implement on both embedded devices and provides two major advantages. First, a nonce is established within a two-way communication process. The TPM and the NFC reader (e.g. a mobile phone) exchange a shared nonce without the need of installing any a priori secrets on the device. Second, the nonce is never transmitted in plaintext so that a potential attacker cannot extract the nonce by simply eavesdropping the communication.

Due to these reasons, we based our NFC nonce-agreement protocol on two NFC standards: the ECMA 385 [9] and ECMA 386 [10]. These standards define security services and protocols for NFC communication. The standards specify several schemes for a secure communication channel and a shared secret service for NFCIP-1-enabled devices. They define cryptographic mechanisms that use the Elliptic Curves Diffie–Hellman (ECDH) protocol for key agreement and the AES algorithm for data encryption and integrity. However, TPM devices which are currently on the market do not support Elliptic-Curve Cryptography (ECC) nor AES encryption. Therefore, we implemented the non-elliptic-curve version of Diffie–Hellman for our NFC demonstrator. Note that the protocol domain parameters such as the prime p, the private keys of both entities s_A and s_B, and the primitive root g need to be known by both entities.

In Fig. 3, the NFC nonce-agreement protocol is shown. The TPM and the mobile phone agree on a shared secret key (nonce) as follows. First, each entity calculates its public key (Q_A and Q_B). Second, the mobile phone sends its public key

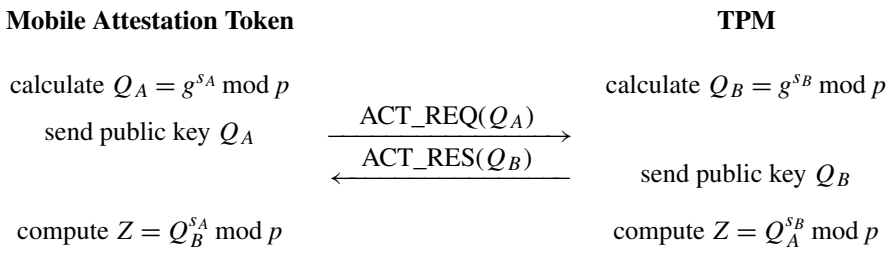

Fig. 3 NFC nonce-agreement protocol between mobile attestation token and TPM

```
TPM_NFC_DH_PARMS
  TPM_STRUCTURE_TAG tag;
  UNIT32 root;
  UINT32 keyLength;
  BYTE* key;
  UINT32 primeLength;
  BYTE* prime;
```

Fig. 4 The TPM_NFC_DH_PARMS data structure holds the information transmitted in the DH scheme

Q_A plus the required parameters p and g to the TPM using the ACT_REQ command according to the ECMA 386 standard. The data is encoded in a byte structure (TPM_NFC_DH_PARMS) which is shown in Fig. 4. Third, the TPM sends the similarly encoded public key Q_B to the mobile phone using the ACT_RES command. In the last step of the key-agreement protocol, both entities compute a shared secret Z according to the DH primitive as specified in 6.2.1 DLSVDP-DH of IEEE 1363 [16], with $SHA-1(Z)$ being the value of TPM_NFC_NONCE.

5.3 Integration in the MAT protocol

The protocol can be easily integrated into the MAP protocol described in Sect. 4. The following steps have to be added. The complete protocol flow including the additional NFC steps is shown in Fig. 2. The additions are drawn in dotted lines.

3. (a) When the MAT and the kiosk touch, the key-agreement protocol as described above is executed and both entities share knowledge of TPM_NFC_NONCE.
   (b) The secret nonce is encrypted by the MAT under the public key of VS and forwarded to the kiosk.
4. Note that TPM_NFC_NONCE is implicitly included in the quote result by the TPM.
6. (a) The Kiosk passes the encrypted nonce on to VS.
7. The trusted bit is only true if the same TPM_NFC_NONCE was received from the MAT, because it is contained in the TPM quote. Note that the TPM with its AIK implicitly guarantees for the authenticity of the nonce origin.

With these changes, our attestation protocol first performs a secure and eavesdropping resistant nonce exchange between TPM and MAT. It then uses this nonce in the quote operation. Therefore the physical proof-of-locality is implicitly guaranteed after completion of the protocol.

6 Implementation details

6.1 Mobile attestation token

We base our MAT prototype implementation on commodity hardware and on platform-independent software. On the Verification Server, Java SE is used. The Verification Server stores reference Known-Good-Values in a relational MySQL database, which is accessed using Hibernate. We supply a GUI tool to allow the user to

Fig. 5 The Mobile Attestation Token software informs the user on the result of the Attestation process in a comprehensible way

collect reference measurements. The Mobile Attestation Token is built as applet for Java ME, MIDP 2.0, extended with JSR 82 (Bluetooth/OBEX support), JSR 75 (PDA profile) and JSR 257 (NFC support). For cryptographic support we use IAIK JCE ME on all hosts. The MAT software is thus compatible to NFC-enabled phones such as the Nokia 6212. Figure 5 shows typical screenshots on the MAT. As no NFC-enabled TPM is currently available in hardware, we simulate the high-level MAT protocol communications using Bluetooth and separately demonstrate the NFC interface.

6.2 Kiosk software platform

On the Kiosk, we currently collect binary measurements, and accept plug-ins for trust property analysis. The TPM can be accessed from Java SE using IAIK jTSS.[6]

To assess the complexity of collecting measurements of a typical kiosk application, we customized a Linux system. Based on November 2009 source files from the Gentoo distribution, we built a restricted base system, added the fvwm2 basic X Window Manager and the Firefox web browser. In the kernel, we included the IMA [33] Linux Security Module (LSM) which performs measurements of all accessed files. We then performed a measured boot and launched the browser afterwards. Note that the exact order and number of programs and libraries loaded depends on external factors like network services or user interactions. The chain-of-trust created by this typically consists of around 520 different IMA measurements. This is of comparable magnitude to the complexity assessment by Lyle and Martin [24], who identified 277 different measurements for a basic web service and about five time as many security-relevant updates over a three-year period. Based on this we believe that several thousand of known-good-values will need to be managed for each kiosk software platform and its updates.

6.3 NFC demonstrator

In order to demonstrate an autonomic and NFC-compatible interface for a TPM, we developed a low-cost NFC prototype. The prototype simply represents a TPM that is assembled on a Printed Circuit Board (PCB). As TPMs available on the market cannot

[6] http://trustedjava.sf.net/.

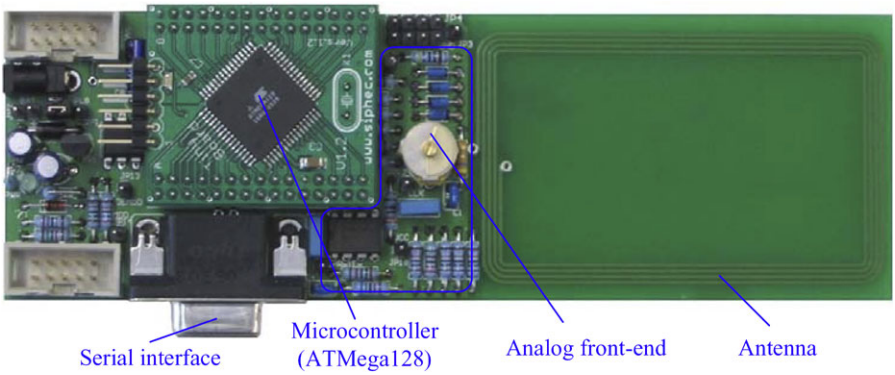

Fig. 6 A picture of the NFC demonstrator printed circuit board

be freely programmed or extended with additional communication interfaces, we use an 8-bit microcontroller in our experiments. Moreover, we integrate an antenna into the PCB which allows an easy access point that can be touched with an NFC-enabled mobile phone. The antenna has been designed according to ISO/IEC 7810 [18] and has a size of a conventional smart card (ID-1 format). Besides the microcontroller and the NFC antenna, the prototype consists of an analog front-end, a clock oscillator, a serial interface, a Joint Test Action Group (JTAG) interface, and a power-supply connector. It operates at 13.56 MHz and has been assembled using discrete components. In Fig. 6, a picture of our prototype is shown.

As a microcontroller, the ATmega128 [1] has been used, which is responsible for managing NFC-reader requests and target responses by following the specification of the NFC Forum. Next to the air interface, the microcontroller is able to communicate with a PC over a serial interface. It furthermore has a JTAG interface for debug control and system programming. The NFC prototype is semi-passive which means that the microcontroller is powered by an external power source while the RF communication is done passively without any signal amplification.

The analog front-end is responsible to transform the analog signals of the NFC reader into digital signals used for the microcontroller and TPM, respectively. Note that the analog front-end and the digital circuit are typically integrated into one piece of silicon as it is in the case of passive RFID tags. The analog front-end is mainly composed of an antenna matching circuit, a rectifier, a voltage regulation unit, a demodulation and a modulation circuit.

The NFC prototype can communicate using several protocol standards. It implements several RFID protocols such as ISO/IEC 15693, ISO/IEC 14443 (types A and B), ISO/IEC 14443-4 and also ISO/IEC 18092. The software is written in C while parts have been implemented in assembly language due to timing constraints. Moreover, it implements a user-command interface that allows easy administration over the serial interface. For our experiments, we have used the ISO/IEC 14443-A [19] protocol standard up to layer 4 using ISO/IEC 7816-4 [17] Application Protocol Data Units (APDU) according to the NFC Forum type 4 tag operation specification [27].

For our experiments, we implemented the key-agreement protocol described in Sect. 5 on our NFC prototype and also on an NFC-compatible mobile phone (Nokia

6212). Both devices are capable of transmitting NDEF messages, which allows an automatic key agreement between the NFC prototype and the mobile phone by simply touching the antenna of the prototype with the mobile phone. After that, we measured the running time of the proposed NFC protocol using an 8-bit oscilloscope. First, the mobile phone sends a request command (REQA) to the NFC prototype which answers with its unique ID (UID) number after an anti-collision and initialization phase. This phase takes about 22.5 ms in our experiments. Second, the mobile phone calculates the public key using a private key and prime size of 768 bits. After that, it sends the generated TPM_NFC_DH_PARAMS structure as a payload of the ACT_REQ command to the NFC prototype. The size of the payload has been 1648 bits which needs about 141.8 ms of transmission time. Note that the time between the first bit transmitted and the last bit received has been measured. Third, the same computation is performed by the NFC prototype which answers with an ACT_RES command using the same payload size needing again 141.8 ms. In our experiments, the entire key-agreement protocol can be performed within one second.

7 Conclusions

In this paper we consider challenges that arise in Remote Attestation scenarios with locally presented computer systems. We conclude that a trusted mobile device, the Mobile Attestation Token, is needed to interact with local computer platforms. It will perform an attestation protocol, report the result to the user, even if the display the user faces cannot be trusted and may be connected to the platform under test.

We extend the previous proposals in this field to provide more scalability considering the limited computational power and memory of mobile devices, and add flexibility by moving the complex state analysis to a trusted third party. Our scheme can be implemented without special-purpose hardware and is not restricted to specific operators. While it does not overcome all complexities of attestation, our scheme allows for full user control over security requirements and trust policies and thus to maintain full self-determination in their trust decisions.

Furthermore, we describe how to add a direct, affordable interface to the TPM. With Near Field Communications, a proof-of-locality is embedded in the attestation process. The presented extension allows us to completely circumvent any malicious software and thus prevent platform-in-the-middle attacks on public available computers.

Acknowledgements The authors thank Manuel Schallar and Herwig Guggi for assisting with the implementation of the Mobile Attestation Token. The work described in this article was supported by the Österreichische Forschungsförderungsgesellschaft through project acTvSM, FIT-IT No. 820848.

References

1. Atmel Corporation (August 2007) 8-bit AVR microcontroller with 128K bytes in-system programmable flash. Available online at http://www.atmel.com/dyn/resources/prod_documents/doc2467.pdf
2. Bangerter E, Djackov M, Sadeghi A-R (2008) A demonstrative ad hoc attestation system. In: Wu T-C, Lei C-L, Rijmen V, Lee D-T (eds) ISC. Lecture notes in computer science, vol 5222. Springer, Berlin, pp 17–30

3. Brickell E, Camenisch J, Chen L (2004) Direct anonymous attestation. In: Proceedings of the 11th ACM conference on computer and communications security, Washington DC, USA, 2004. ACM, New York, pp 132–145
4. Cáceres R, Carter C, Narayanaswami C, Raghunath M (2005) Reincarnating PCs with portable soulpads. In: Proceedings of the 3rd international conference on mobile systems, applications, and services, Seattle, Washington, 2005. ACM, New York, pp 65–78
5. Chen L, Landfermann R, Löhr H, Rohe M, Sadeghi A-R, Stüble C (2006) A protocol for property-based attestation. In: STC '06: Proceedings of the first ACM workshop on scalable trusted computing
6. Coker G, Guttman J, Loscocco P, Sheehy J, Sniffen B (2008) Attestation: Evidence and trust. In: ICICS'08: Proceedings of the 10th international conference on information and communications security. Springer, Berlin, pp 1–18
7. Grawrock D (2006) The intel safer computing initiative. Intel Press, Hillsboro. ISBN 0-9764832-6-2
8. Grawrock D (2009) Dynamics of a trusted platform: a building block approach. Intel Press, Hillsboro. ISBN 978-1934053171
9. ECMA International (December 2008) ECMA standard 385-2008: NFC-SEC: NFCIP-1 security services and protocol
10. ECMA International (December 2008) ECMA Standard 386-2008: NFC-SEC-01: NFC-SEC cryptography standard using ECDH and AES
11. EMSCB Project Consortium (2004) The European multilaterally secure computing base (EMSCB) project. http://www.emscb.org/
12. England P (2008) Practical techniques for operating system attestation. In: Trust '08: Proceedings of the 1st international conference on trusted computing and trust in information technologies. Springer, Berlin, pp 1–13
13. Garfinkel T, Pfaff B, Chow J, Rosenblum M, Boneh D (2003) Terra: A virtual machine-based platform for trusted computing. In: Proceedings of the 19th symposium on operating system principles (SOSP 2003). ACM Press, New York, pp 193–206
14. Garriss S, Cáceres R, Berger S, Sailer R, van Doorn L, Zhang X (2008) Trustworthy and personalized computing on public kiosks. In: Grunwald D, Han R, de Lara E, Ellis CS (eds) MobiSys. ACM, New York, pp 199–210
15. Hancke G (2008) Eavesdropping attacks on high-frequency RFID tokens. In: Workshop on RFID security 2008 (RFIDSec08), July 9–11, Budapest, Hungary, Vol RFIDsec 2008, pp 100–113
16. IEEE (2000) IEEE standard 1363-2000: IEEE standard specifications for public-key cryptography. Available online at http://ieeexplore.ieee.org/servlet/opac?punumber=7168
17. International Organisation for Standardization (ISO) (1995) ISO/IEC 7816-4: Information technology—identification cards—integrated circuit(s) cards with contacts—Part 4: Interindustry commands for interchange. Available online at http://www.iso.org
18. International Organisation for Standardization (ISO) (2003) ISO/IEC 7810: Identification cards—Physical characteristics
19. International Organization for Standardization (ISO) (2000) ISO/IEC 14443: Identification cards—Contactless integrated circuit(s) cards—proximity cards
20. Kauer B (2007) Oslo: improving the security of trusted computing. In: SS'07: Proceedings of 16th USENIX security symposium, Berkeley, CA, USA, 2007. USENIX Association, pp 1–9
21. Kühn U, Selhorst M, Stüble C (2007) Realizing property-based attestation and sealing with commonly available hard- and software. In STC '07: Proceedings of the 2007 ACM workshop on scalable trusted computing
22. Li F, Wang W, Ma J, Ding Z (2008) Enhanced architecture of TPM. In: Young computer scientists, 2008. ICYCS 2008. The 9th international conference for, pp 1532–1537
23. Lindner F (2007) Toying with barcodes. In: 24th chaos communication congress
24. Lyle J, Martin A (2009) On the feasibility of remote attestation for web services. In: Proceedings of the 2009 international conference on computational science and engineering, vol 03. IEEE, New York, pp 283–288
25. McCune J, Perrig A, Reiter M (2005) Seeing-is-believing: Using camera phones for human-verifiable authentication. In: Security and privacy, 2005 IEEE symposium on, pp 110–124
26. McCune JM, Perrig A, Seshadri A, van Doorn L (August 2007) Turtles all the way down: Research challenges in user-based attestation. In: Proceedings of the workshop on hot topics in security (HotSec)
27. NFC Forum (March 2007) NFC forum type 4 tag operation—technical specification
28. OpenTC Project Consortium (2005–2009) The open trusted computing (OpenTC) project. http://www.opentc.net/

29. Oprea A, Balfanz D, Durfee G, Smetters DK (2004) Securing a remote terminal application with a mobile trusted device. In: ACSAC
30. Parno B (2008) Bootstrapping trust in a "trusted" platform. In: Proceedings of the 3rd conference on hot topics in security, San Jose, CA, 2008. USENIX Association, pp 1–6
31. Pirker M, Toegl R, Hein D, Danner P (2009) A PrivacyCA for anonymity and trust. In: Chen L, Mitchell CJ, Andrew M (eds) Trust '09: Proceedings of the 2nd international conference on trusted computing. Lecture notes in computer science, vol 5471. Springer, Berlin
32. Sadeghi A-R, Stüble C (2004) Property-based attestation for computing platforms: Caring about properties, not mechanisms. In: Hempelmann C, Raskin V (eds) NSPW. ACM, New York, pp 67–77
33. Sailer R, Zhang X, Jaeger T, van Doorn L (2004) Design and implementation of a TCG-based integrity measurement architecture. In: Proceedings of the 13th conference on USENIX security symposium, vol 13, San Diego, CA, 2004. USENIX Association, pp 16–16
34. Sharp R, Scott J, Beresford AR (2006) Secure mobile computing via public terminals. In: Fishkin KP, Schiele B, Nixon P, Quigley AJ (eds) Pervasive. Lecture notes in computer science, vol 3968. Springer, Berlin, pp 238–253
35. Trusted Computing Group (2007) TCG software stack specification, version 1.2 errata a. https://www.trustedcomputinggroup.org/specs/TSS/
36. Trusted Computing Group (2007) TCG TPM specification version 1.2 revision 103. https://www.trustedcomputinggroup.org/specs/TPM/