Touch'n Trust: An NFC-Enabled Trusted Platform Module

Michael Hutter and Ronald Toegl
Institute for Applied Information Processing and Communications (IAIK)
Graz University of Technology, Inffeldgasse 16a, A-8010 Graz, Austria
Email:{michael.hutter,ronald.toegl}@iaik.tugraz.at

***Abstract*—Instant and ubiquitous access to devices such as public terminals raises several security concerns in terms of confidentiality and trust. While Trusted Computing introduces advanced security mechanisms into terminal hardware, there is often no convenient way to help users decide on the trustworthiness of a device. To overcome this issue, Near Field Communication (NFC) can be used to leverage the trusted-computing protocol of remote attestation. Here, NFC helps user to intuitively establish a communication between local devices. In this article, we propose an NFC-enabled Trusted Platform Module (TPM) architecture that allows users to verify the security status of public terminals. For this, we introduce an autonomic and low-cost NFC-compatible interface to the TPM to create a direct trusted channel. Users can access the TPM with NFC-enabled devices, which have become widely available in the form of smart phones. Elliptic-curve cryptography provides efficient signing and verifying of the security-status report. Furthermore, we implemented an NFC-enabled TPM platform as a proof-of-concept demonstrator and show that a trust decision can be realized with commodity mobile phones. It shows that an NFC-enabled TPM can effectively help to overcome confidentiality issues in common public-terminal applications.**

***Keywords*-Trusted Computing; RFID Security; Near Field Communication; NFC; ECDSA; Remote Attestation.**

I. Introduction

During the last decade, mobile technology has grown significantly offering many applications including payment, ticketing, and access control. Especially Near-Field Communication (NFC) has become widely available on the market offering convenient services by simply touching NFC-enabled objects in the proximity with commodity mobile phones. It is obvious that such ubiquitous communication raises concerns on the security and on the privacy, especially as NFC services are able to pinpoint the location of their users. Therefore, for users it would be desirable to only use NFC services that are worth of being entrusted with sensitive or personal information. In this article, which extends [1], we present a method to overcome confidentiality and trust issues in security-related NFC applications by taking advantage of Trusted Computing mechanisms.

Trusted Computing has been designed to provide services to establish trust in Internet environments. One such service, as devised by the Trusted Computing Group (TCG) industry consortium, is *remote attestation*. It achieves trust decisions between hosts. A TPM is therefore used to measure the integrity and confidentiality guaranteeing status of a targeted host platform. This state information must then be reported to the user who decides if the platform can be trusted or not. Unfortunately, the cryptographic protocols that actually perform the attestation are not suited for local and interactive service scenarios. They neither provide a human intelligible conveyance of the status report nor an intuitive identification of the targeted host platform. In a mobile user scenario, small and portable devices such as mobile phones or PDAs can help perform the attestation protocol reporting the status of the targeted platform to the user [11], [29], [47], [48]. It has also been established that to provide such a service, a trusted channel between the user and the platform-integrated TPM is mandatory [33].

In this article, we propose to integrate both an NFC interface and a TPM into a single hardware component. This integration provides a secure trust decision to the user by guaranteeing the physical presence of the user through NFC and offering the direct channel to the TPM by the combination of both modules in one piece of silicon. The proposed architecture leverages the remote attestation protocol so that a mobile phone can be used to establish trust to a public terminal in the proximity. We improve on a previously reported proposal [48] by moving the trust decision into the mobile device. This is enabled by substantial optimizations that include an advanced terminal hardware and software platform and a more efficient cryptographic protocol. To this end, we propose to use elliptic curve cryptography (ECC) to increase the computation and communication performance. A virtualization-based software architecture allows for compact and expressive state descriptions. As a proof-of-concept, we implemented an NFC-enabled TPM prototype that performs remote attestation using the elliptic curve digital signature algorithm (ECDSA). We further give performance results of our practical experiments, both in software and hardware.

The remainder of this article is structured as follows. In Section II, we give a short introduction into the NFC technology. Section III introduces Trusted Computing and the Trusted Platform Module. Section IV describes the proposed architecture that integrates an NFC-interface inside the TPM module. We present implementation details and performance results in Section V. The paper concludes in Section VI.

II. NFC Environments

NFC is a wireless communication technology that provides a platform for many applications such as mobile payment, ticketing, and access control. One key feature of NFC is the simple acquisition of data 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 a frequency of 13.56 MHz. 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 modulation, coding, frame formats, data rates, and also the application-specific transport protocols used. The typical operating distance between two NFC devices is only a few centimeters (up to 10 cm). Thus, installing an NFC device (passively or actively powered) to a fixed location can provide evidence whether a mobile NFC device (or its user) has been at that location or not. 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 therefore mimics the natural principle of communication between two locally present entities.

There already exist many mobile devices that support NFC. Typical examples are the 6131, 6212, and 3220 mobile phones from Nokia, the *SGH-X700* or *D500E* from Samsung, the *my700X* from SAGEM, the *L7* from Motorola, or the *T80* from Benq. NFC will be also an integral part of the next generation of smart-phones. Nokia's *C7* smartphone already includes an NFC chip that can be activated by a software update in the first quarter of 2011. Another example is the *Nexus S* smart-phone from Google which has been available since December 2010. There have already been many companies that use that technology to provide different *touch and go* applications, *e.g.*, the *Touch and Travel* project of the German railway system and the mobile operator Vodafone, the e-ticket and e-payment services of Japan's mobile operator KDDI or NTT DoCoMo, or the Bay Area Rapid Transit (BART) system in the San Francisco Bay area.

The use and integration of NFC into next generation devices has been largely pushed and promoted by the NFC Forum since 2004. The NFC Forum is an association of about 140 industry partners such as NXP Semiconductors, Sony, Nokia, Microsoft, MasterCard, NEC, Visa, and Samsung. They encourage the development of new standards and products to ensure the interoperability of NFC-enabled devices. They defined four different types of NFC-Forum compatible devices. All types are based on the contact-less smart-card standard ISO 14443 [19] or FeliCa. Type 1 tags provide only simple functionalities such as writing of data into the memory while more complex type 4 tags typically involve also security features and support ISO/IEC 7816-4 Application Protocol Data Units (APDU).

However, as soon as entities use NFC services and establish a connection to terminals with their mobile phones, the questions of integrity and confidentiality rise inevitably. Especially in security-related applications like mobile payment and ticketing, cryptographic services are needed that provide a decision mechanism to the user if the connected device can be trusted or not. Compromised devices pose a serious thread that might extract secret information, perform unwanted payment transactions or behave untrustworthy in some other way.

III. Trusted Computing enabled Terminals

A. Trusted Platform Module

Trusted computing offers services to make trust decisions by integrating a so-called Trusted Platform Module (TPM) [45] into target machines, *e.g.*, public NFC terminals. The Trusted Computing Group (TCG) has specified the TPM for general purpose computer systems. Similar to a smart card, the TPM features cryptographic primitives but it is physically bound to its host device. A tamper-resilient integrated circuit contains implementations for public-key cryptography, key generation, cryptographic hashing, and random-number generation and provides therefore a root of trust.

In particular, the TPM implements high-level functionalities such as reporting the current system configuration and providing evidence of the integrity and authenticity of this measurement. This service is also known as *Remote Attestation*. During the remote attestation process, the TPM receives hashes of several system-state descriptors and stores the hashes in dedicated Platform Configuration Registers (PCRs) located in the TPM. In fact, a PCR with index i in state t is extended with input x by setting

$$PCR_i^{t+1} = \text{SHA1}(PCR_i^t || x).$$

The basic operation of a TPM is as follows. Before executable code is invoked, a hash value of the code is computed and stored in a PCR. Ultimately, if all components from the Basic Input/Output System (BIOS) up to a specific application are measured, the exact configuration of the platform is mapped to PCR values. This property makes it impossible to hide a malicious program on a thus protected computer. If such a system state fulfills the given security or policy requirements, we refer to the system state as a *trusted state*.

The TPM can also *bind* data to the platform by encrypting it with a *non-migratable* key, which never leaves the TPM's protection. An extension to this is *sealing*, where a key may only be used with a specific PCR configuration. Thus, decryption of sealed data can be restricted to a trusted state of the computer. TPMs also provide a limited amount of non-volatile memory (NV-RAM) to store user- or owner-supplied information.

The TPM is capable of signing the current values of the PCRs together with a supplied nonce. This is called the *Quote* operation, which is the core operation in the *Remote Attestation* protocol [8], [40], [44]. To protect the platform owner's privacy, a pseudonym identity is used: an *Attestation Identity Key (AIK)*. The authenticity of an AIK can be certified either by an on-line trusted third party, called PrivacyCA [34] or by applying the group-signature-based DAA scheme [7], for instance. Then, a remote verifier may analyze the Quote result and decide whether to trust the given configuration or not.

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. 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) [46] which covers operating system and applications support.

B. Platform Virtualization for Attestation

Virtualization is a methodology of dividing the resources of a computer into multiple execution environments, by applying concepts such as time-sharing [43], hardware and software partitioning, machine simulation or emulation. Hardware architectures can be designed to offer complete virtualization [38] in hardware and then host unmodified operating systems in parallel. Only recently, the PC platform has been modified accordingly (see [2] for an overview).

Commonly, virtualization is controlled by a singleton *hypervisor*, a superior control entity which directly runs on the hardware and manages it exclusively. It enables the creation, execution and hibernation of isolated *partitions*, each hosting a guest operating system and the virtual applications building on it.

Such a system provides multiple isolation layers: Standard processor privilege rings and memory paging protect processes executing within a virtualization. Hardware support for monitoring all privileged CPU instructions enables the hypervisor to transparently isolate virtualization instances from each other. Finally, the chip-set is able to block direct memory accesses (DMA) of devices to defined physical memory areas, thus allowing the hypervisor to control device I/O.

Modern platforms from AMD [3] and Intel [12] extend the basic TCG model of a *static* chain-of-trust anchored in a hardware reboot. They provide the option of a *dynamic* switch to a trusted system state. In this paper we focus on Intel's *Trusted Execution Technology* (TXT), which we build our implementation on. Similar functionality is provided by AMD's Secure Virtual Machine (SVM).

A so-called *late launch* is initiated by the special Intel TXT CPU instruction `GETSEC[SENTER]`. It stops all processing cores except one. The chip-set locks all memory to prevent outside modification by DMA devices. A special Intel-provided and cryptographically signed *Authenticated Code Module* (ACM) starts a fresh chain-of-trust after setting the platform into a well-defined state. This provides a *Dynamic Root of Trust for Measurement* (DRTM).

Subsequently, a *Measured Launch Environment* (MLE) [18] is first measured and then executed. Piece-by-piece the MLE decides which system resources to unlock and thus cautiously restores normal platform operation. The platform remembers that she is running in "secrets" mode and automatically enforces memory scrubbing whenever a system (re)boot is initiated.

The ACM is also capable of enforcing specific *Launch Control Policies* (LCPs). Here, the ACM measures the MLE and compares it with the trusted LCP stored in the non-volatile memory of the TPM. Changes to the LCP can only be authorized by the TPM owner. Any other, not authorized software configuration is not allowed to continue; the ACM will reset the platform. This mechanism thus allows to perform a secure boot into a trusted state. Virtualization also allows to measure complete file system images that contain virtual applications. This leads to deterministic, yet expressive PCR values.

C. Remote Attestation over NFC

Attestation is useful to improve the security for a number of computing services, including not only remote but 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. 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. In addition, giving voters a method to validate that electronic voting machines have not been tampered might assist to add trust to a poll's outcome.

Public terminals are typically located at shops, in hotel lobbies, transportation terminals, or Internet cafés. They provide different services to users like Internet access via Web browsers or ticket-vending services. These terminals are publicly available and can be accessed by users several times always pretending a legitimate user. Possible attackers are therefore assumed to have full access over the software running on the terminals. As a result, the terminal can not be trusted. The users get no guarantee about the confidentiality and privacy status of the terminal they would like to use.

With TPM-based attestation, trust can be established between users and terminals. There exist several proposals for that. McCune et al. [29] pointed out that it would be desirable to equip the user with a simple, ideal and axiomatically trustworthy device. It would then indicate the security of a device to the user. Molnar et al. [30] describe an RFID-based solution. They proposed to integrate the remote attestation protocol into RFID readers to allow the verification of the privacy status of the reader to existing tags (or users) in the field. Li et al. [25] proposed to secure mobile-payment applications with remote attestation. They present practical results of an attestation scenario using NFC mobile phones. A similar approach has been presented by Garriss et al. [11]. They designed a protocol by which a mobile phone can determine the integrity of running software on a terminal (kiosk computer).

However, many of the proposed architectures suffer in the fact that the TCG's attestation protocol does not guarantee that the TPM is still located within the machine the user faces. This allows possible machine-in-the-middle attacks where an attacker establishes an indirect link between the user and the TPM over a distrusted channel.

IV. The NFC-Enabled TPM Architecture

In the following section, we briefly outline our design for an NFC-enabled TPM architecture. In contrast to existing publications, we propose to integrate the functionality of a low-cost passive RFID interface into the TPM. This allows users to remotely audit the privacy status of the terminal (target) using a conventional NFC device (initiator) by ensuring a direct channel between the user and the TPM. Figure 1 shows a schematic view of our proposed architecture. It shows the TPM device with its components such as a CPU, I/O interface, voltage regulator, memory, and cryptographic components like SHA-1, RSA/ECC, key generation, and a random number generation (RNG). Additionally, an RFID front-end is connected to the internal bus. It is composed of a digital part which handles the NFC protocol (ISO 14443 or ISO 18092) and an analog part that is mainly responsible to modulate and demodulate the signals of the air interface. Next to the I/O and power interface, the TPM has two additional connections for an external antenna that can be connected or printed on the main board of the terminal.

In general, passive RFID/NFC devices are designed to meet low-resource requirements. They draw their power of the air from the reader and therefore consume only a few microwatts of power to provide a certain reading range. Furthermore, they meet low area requirements so that they can be produced in a large scale with low costs. In fact, most RFID tags can nowadays be produced with costs below 10 cents.

TPMs, in contrast, have an active power supply and provide many resources such as cryptographic engines, CPU, and (in comparison) large memory units. These resources can actually be reused be the RFID/NFC front-end which lowers the requirements of integrating NFC into TPM significantly. The controlling of the protocol, for example, can be handled by the CPU of the TPM and memory can be shared with the RFID/NFC component over the internal bus architecture as well.

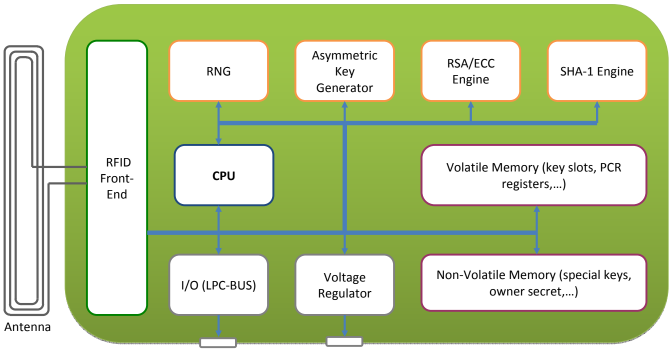


Figure 1. Schematic of the NFC-Enabled TPM architecture.

By integrating an NFC interface to the TPM, we followed the idea of Parno [33] who recommended to integrate a special-purpose hardware interface to the TPM to establish a direct link between the user and the TPM so that human inter-actors can themselves establish the proximity of the attesting machine. The integration of the NFC interface to the TPM has thereby several advantages.

First, a guarantee is given that the targeted terminal is still in the proximity of the user, as the user performs the touching operation in person.

Second, it gives a guarantee that the TPM is located in the proximity because the NFC module is physically connected and integrated into the TPM and the operational range of NFC is theoretical limited by the coupling property of the magnetic near-field of the reader. Note that RFID standards define a practical reading range of about 10 centimeters. This makes attacks such as machine-in-the-middle attacks much harder to perform.

Third, our proposal of integrating an NFC interface into the TPM allows users to verify the integrity of public terminals with their mobile phone. By simply touching the antenna of the terminal with their phone, an application can be automatically launched that shows the status of the trust decision on a user-friendly and familiar display output. In addition, the trust decision can be also signalized by a beep, ringing tone, or vibration which makes the application more applicable and comfortable in certain environments.

In the following, we describe the implemented trusted computing protocol of remote attestation that can be used to provide such a service.

A. Public Key Infrastructure

To support our software architecture, a Public Key Infrastructure (PKI) is needed. A PKI represents a system for binding a public key to a specific and distinct user or device identity by means of digital certificates. Certificates are signed by a Certification Authority after a Registration Authority established the identity and Revokation mechanisms blacklist compromised keys.

In our scheme a PrivacyCA acts as trusted third party for all participants. The AIK certificates it issues are created during the registration process of the unique terminal hardware platform. It ensures that a real hardware TPM is present on the registered hardware, and that the private part of each AIK will never be exposed by the TPM. Currently only Infineon TPMs can be verified because only they provide certificates for their Endorsement Keys.

Therefore, for every new terminal that is added to the network it is necessary that its TPM has been activated and the ownership has been taken. In the following we describe the registration process. At first, the identity of the hardware platform is established by qualified personnel. After the creation of the TPM-based AIK public/private key pair, the public part is sent to the PrivacyCA server together with the Endorsement Key certificate. The PrivacyCA server validates and analyses the EK certificate and decides whether to trust the TPM or not. If the server believes the TPM to be authentic, the server then certifies the AIK and encrypts the fresh certificate with the public Endorsement Key of the TPM. This is returned to the platform and can only be decrypted there. The AIK certificate is permanently stored at the PrivacyCA to allow a later revocation of this certificate.

B. The Remote Attestation Protocol

The first step in the attestation protocol is to generate a nonce N_0 with the NFC-enabled mobile phone which acts as a reader device (initiator). As soon as the mobile phone touches the RF antenna of the terminal, the nonce is transmitted over the air interface within a configuration challenge (the nonce serves as fresh data to avoid replay attacks). After that, the public terminal (target) responds with the Quote of its currently recorded terminal state. The Quote, *i.e.*, $Sig_{AIK}(PCR_n..PCR_m, N_0)$, $1 \leq n \leq m \leq 24$, the signature over the selected PCR registers under an Attestation Identity Key AIK of the TPM, is then returned to the mobile phone. The protocol flow is shown in Figure 2. Note that the figure shows only a very compact protocol flow that neither includes issues of key management nor the handling of certificates and trusted third party (TTP) services.

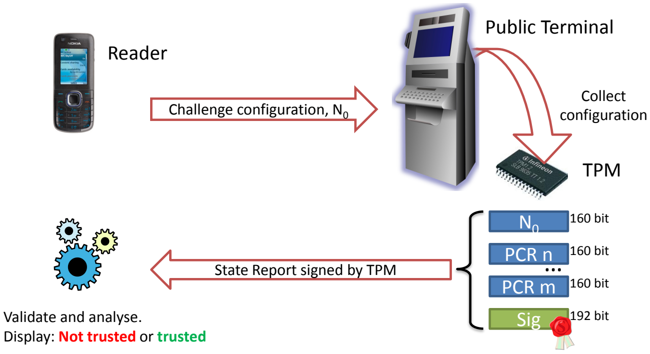


Figure 2. The compact NFC attestation protocol.

In order to sign the PCRs of the TPM prototype, we propose the use of elliptic curve cryptography (ECC). In contrast to other public-key primitives like RSA, ECC has gathered much attention due to the use of shorter key sizes. The computation time and especially the communication time over the air interface can be significantly improved by providing the same cryptographic strength. For instance, the strength of a 2 048 bit RSA key can be compared with that of a 224 bit ECC key.

Due to these reasons, we propose to use ECDSA to sign the data. The use of ECDSA has several advantages. First, the protocol has been standardized by several organizations such as ANSI [4], IEEE [16], NIST [32], and ISO/IEC [20]. Second, there already exist public-key infrastructures that support this algorithm for signing and verifying data and X.509 certificates [21].

The algorithm for generating digital signatures is shown in Algorithm 1. It takes a message m as input (containing N_0 in the TPM-Quote data structure) and outputs the digital signature (r, s) of that message. The private key d is securely stored in non-volatile memory. The most time consuming operation in ECDSA is the elliptic curve (EC) point multiplication $[k]P$ (line 2 in Algorithm 1). It takes more that 80 % of the total execution time. For this, a randomly generated value k (ephemeral key) is multiplied with a point P on an elliptic curve. The x-coordinate of the resulting point is then used further in the signing process. Next to that operation, a message digest algorithm (SHA-1) is used to hash the input message (line 4). The final signing process (line 5) needs several finite-field operations such as modular addition, modular multiplication, and modular inversion.

Algorithm 1 Signature generation using ECDSA

```
Require: Domain parameters, private key $d$, message $m$.
Ensure: Signature $(r, s)$
1: Select $k \in [1, n-1]$
2: Compute $[k]P = (x_1, y_1)$, convert $x_1$ to an integer $\bar{x}_1$
3: Compute $r = \bar{x}_1 \pmod{n}$. If $r = 0$ then go back to 1.
4: Compute $e = SHA1(m)$.
5: Compute $s = k^{-1}(e + dr) \pmod{n}$. If $s = 0$ then go
   back to step 1.
6: Return $(r, s)$
```

An ECDSA digital signature can be verified by the verification algorithm shown in Figure 2. First, the input signature (r, s) is verified to be in $[0, n-1]$ (line 1). After that, the hash of the message m is calculated (line 2). Line 3-4 show the calculation of the intermediate variables w, u_1, and u_2. In line 5, two point multiplications have to be performed resulting in the elliptic-curve point X. The signature is valid if the relation $v = r$ holds, otherwise it is rejected.

Algorithm 2 Signature verification using ECDSA

```
Require: Domain parameters, public key $Q$, message $m$,
    signature $(r, s)$.
Ensure: Accept or Reject of $(r, s)$
1: Verify that $r, s \in [0, n-1]$.
2: Compute $e = SHA1(m)$.
3: Compute $w = s^{-1} \pmod{n}$.
4: Compute $u_1 = ew \pmod{n}$ and $u_2 = rw \pmod{n}$.
5: Compute $X = (x_1, y_1) = u_1 P + u_2 Q$.
6: If $X = \infty$ then return.
7: Compute $v = \bar{x}_1 \pmod{n}$.
8: If $v = r$ then *accept* else $reject$.
```

If the signature has been validated successfully, the Quote information is compared to a list of known-good PCR values. A quote is only accepted if the state report contains only trusted values.

V. Implementation Results

In the following, we present results of an implementation of the proposed system architecture. First, we describe the implementation of a public terminal that runs on an attestation-friendly software platform. Second, we describe an NFC-enabled TPM prototype that can be touched by NFC-enabled mobile phones. Third, we show an attestation scenario example and give results about the implementation's performance.

A. The Public Terminal

As a public terminal, we used an attestation-friendly hardware platform that is based on the Intel Trusted Execution Technology (TXT). We measure and enforce the terminal integrity with the acTvSM software platform [35], [36], [49]. It relies on a TPM to provide basic Trusted Computing services such as secure storage of software measurements and on hardware-based virtualization to execute programs in a trusted environment. The terminal application is, together with its operating system, contained in an image file, which is measured before execution. From within the virtual machine, we can access the TPM using IAIK jTSS [37] to retrieve the Quote that reflects the system state.

1) Software Components: Secure boot is accomplished by using a standard boot-loader GRUB [10] along with `SINIT` and `tboot` [17]. `SINIT` is Intel's implementation of an ACM, while `tboot` is Intel's prototype implementation of an MLE. Upon boot GRUB loads SINIT, tboot, the kernel and its `initramfs` into memory and executes tboot, which sets up the ACM and then late-launches into it. The authenticity and integrity of the ACM code is guaranteed under an Intel private key, of which the public part is hardwired into the chip-set. The ACM's task is then to measure the tboot binary and compare it to the LCP. Tboot takes over and continues to measure the kernel and `initramfs` and compares them against the VLP. Once the integrity of the kernel and its environment has been assured, control is passed to it and the standard boot process continues. Customized 64-bit ports of tools from IBM's *TPM-utils* [15] provide the PCR extend and unsealing capabilities in the initial ram-disk (`initramfs`) environment.

In our architecture, we use a customized Linux operating system augmented with the *Kernel-based Virtual Machine* (KVM) [23], [24] hypervisor module. KVM can run multiple virtual machines on x86 CPUs equipped with virtualization mode extensions. It extends the Linux Kernel to offer, besides the existing Kernel and User modes, an additional Guest process mode. Each virtual machine offers private virtualized hardware like a network card, hard disk, graphics adapter, etc. Those virtual devices are forwarded to *QEMU* [6], a fast software emulator. QEMU can emulate all standard hardware devices of a x86 platform, including one or several processors. For the Base system, we use packages from the x86_64 *Debian Linux* lenny release [41]. It acts as the host for the virtualization partitions. To support current Trusted Computing and virtualization hardware we need to add selected packages from the Debian testing tree. For example, only Linux kernels 2.6.32 or newer integrate Intel TXT support and drivers for chip-sets that implement it. Scripts for installation, initial ram-disk management and rebuilding of the Base System image are taken from Debian and customized to our needs. The system bootstrap scripts for creation of distributable and boot-able CDs for initial installation are taken from GRML Linux [39], a distribution specialized for system administrators.

2) Application Image Management: We use a complex disk layout with different file systems to create a measurable platform.

A first partition contains a read-write file-system hosting all the components necessary for the platform boot process. This encompasses the boot-loader, `tboot`, `SINIT` and Linux kernel plus associated initramfs images. The remainder of the harddisk storage is allocated as a Logical Volume Manager (LVM) [27] dynamically managed space, which is assigned to a single LVM volume group.

The LVM managed volume group contains both plain-text and encrypted logical volumes (LVs). These block devices are transparently encrypted with Linux kernel's `dm-crypt` subsystem. Each LV contains either a file system for use on the platform, or raw storage which is assigned to, and managed by, virtual partitions. `dm-crypt` encrypts block devices with symmetric AES keys, called master-keys. Each encrypted LV contains a Linux Unified Key Setup (LUKS) [26] partition header, which is responsible for key management.

As a running Linux system requires some writable file system, the root "`/`" file system of the platform is assembled from multiple layers with an in-memory `tmpfs` to provide writable, but ephemeral storage.

Each application-specific logical volume contains one virtual partition application image, *i.e.*, a Terminal front-end which offers services to users.

Note that due to these structured file systems, complete measurements of the overall system configurations are composed of only a few hashes from well-known (read-only) software images and can therefore be compared easily with reference values in NFC-enabled mobile phones.

B. The NFC-Enabled TPM Prototype

In order to demonstrate an autonomic and NFC-compatible TPM that runs on the public terminal, we developed a low-cost NFC prototype. The prototype simply represents a TPM that is assembled on a Printed Circuit Board (PCB). Instead of conventional TPM modules, which are sealed and protected against modifications, we used an 8-bit microcontroller from Atmel (ATmega128) for our demonstration that can be freely programmed over a standard JTAG interface. The microcontroller has 128 kB of Flash memory, 4 kB of RAM, and operates at 13.56 MHz. Furthermore, the microcontroller is directly connected to an analog antenna circuit that has been also integrated on the PCB. It has been designed according to ISO/IEC 7810 and has a size of a conventional smart card (ID-1 format). This interface provides an easy access point for NFC-enabled devices. For debugging purposes, there also exists a serial interface on the PCB that allows a communication between the TPM prototype and a PC. In Figure 3, a picture of our NFC prototype is shown where it gets touched by an NFC-enabled mobile phone.

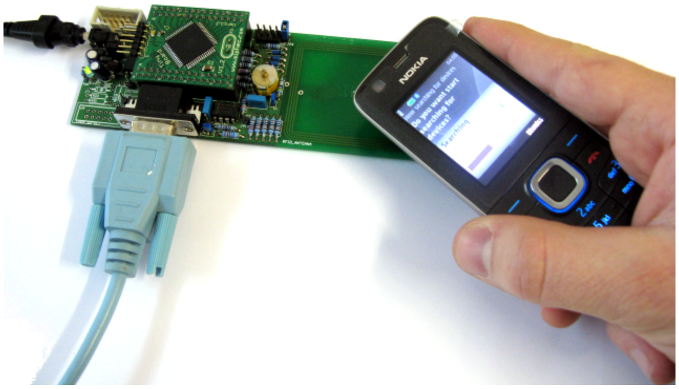


Figure 3. The NFC-enabled TPM prototype.

For RFID and NFC communication, our TPM prototype implements several protocol standards. It implements RFID protocols such as ISO/IEC 15693, ISO/IEC 14443 (type A), 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 Application Protocol Data Units (APDU) according to the NFC Forum type 4 tag operation specification [31].

In order to transmit and receive data from and to an NFC-enabled device, we encapsulated the payload using the specified NFC Data Exchange Format (NDEF). In particular, NDEF can be used by NFC-enabled mobile phones to allow an automatic launch of applications when the phone gets close to our prototype. For this, we support several NDEF records and allow user specific information to be transmitted to a mobile phone that get close to our prototype.

In order to sign the PCRs of the TPM prototype, we implemented ECDSA according to the recommendations of the National Institute of Standards and Technology (NIST). The implementation is based on the digital-signature standard [32] and uses the smallest recommended elliptic curve that is 192 bits for prime-field arithmetic. As a point-multiplication method (see line 2 in Algorithm 1), we decided to implement the improved Montgomery ladder proposed by Izu, Möller, and Takagi [22]. This method is shown in Algorithm 3 and performs a point addition and doubling operation in every Montgomery-ladder loop iteration to multiply the ephemeral key k with the fixed base point P of the elliptic curve. We performed all computations with (homogeneous) projective coordinates (the formulae used are given in [14]) and applied a coordinate randomization technique according to Coron [9]. For this, we randomized every intermediate value during the scalar multiplication using a random value λ (line 1 in Algorithm 3). This makes our implementation more resistant to side-channel attacks [28] which try to reveal the ephemeral key used during the signature generation.

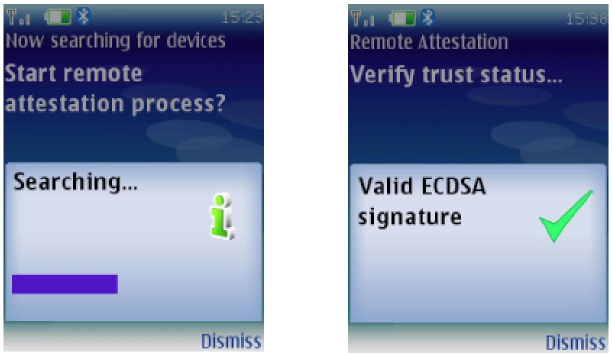
Algorithm 3 Montgomery ladder according to [22].

```
Require: $P = (P_x, P_y) \in E(\mathbb{F}_{p192}), k \in [1, 2^{192} - 1]$,
    random $\lambda$, $k \geq 2^{190}$.
Ensure: $Q = kP$, where $Q = (x, y) \in E(\mathbb{F}_{p192})$
1: $Q_0 = (X_0, Z_0) \leftarrow (\lambda P_x, \lambda)$.
2: $Q_1 = (X_1, Z_1) \leftarrow Dbl(P)$.
3: for $i = 190$ downto $0$ do
4:     $(Q_{k_i \otimes 1}, Q_{k_i}) \leftarrow ECCAddDbl(Q_{k_i}, Q_{k_i \otimes 1})$
5: end for
6: $x \leftarrow x(Q_0) = X_0 \cdot Z_0^{-1} \pmod{p}$.
7: Return $(x)$.
```

As shown in Algorithm 3, we combined both group operations which are point addition and point doubling to one operation ($ECCAddDbl$). The entire operation needs four variables to store the two projective curve points (X_0, Z_0, X_1, Z_1) and seven intermediate variables to maintain the intermediate coordinates. Note that no y-coordinates have to be maintained during point multiplication due to the use of the Montgomery ladder. In addition, for digitally signing with ECDSA it is not necessary to recover the y-coordinate because only the final x-coordinate is used after the scalar multiplication. All finite-field operations have been implemented in C except the modular multiplication algorithm which was implemented in Assembly language due to performance reasons. The multiplication algorithm uses a product scanning form (Comba multiplication) and applies the fast NIST reduction algorithm to reduce the result [13]. For modular inversion, we implemented the binary algorithm which has been adapted from the extended Greatest Common Divisor (GCD) algorithm from [13], [42]. As a modular reduction method, we applied the algorithm proposed by Barrett [5].

C. The NFC-Attestation Scenario

In the NFC attestation scenario, an NFC-enabled device is used to touch the antenna of the TPM prototype. For this scenario, we used the NFC edition of the Nokia 6212 mobile phone. It is shipped with an integrated RFID-reader chip that allows touching of NFC-enabled objects in the near proximity. Using the Nokia Software Developer Kid (SDK), we implemented a Java J2ME Midlet that runs on the phone. We implemented three threads: `SearchThread`, `SignatureGeneratingThread`, and `SignatureVerifyingThread`. The `SearchThread` handles the detection of passive NFC devices in the field. If our NFC prototype gets touched by the phone, it is detected by the `DiscoveryManager` and an `ISO14443Connection` is established by the Midlet. After that, the Midlet sends an ISO 7816-4 APDU to the prototype to start the attestation process. The prototype signs the PCR values together with the nonce N_0. For this, we implemented the same algorithm of ECDSA in a Java Midlet allowing signing and also verifying of digital signatures. The used ECC parameters such as the type of elliptic curve, the curve parameters (a and b), or the base point P have been fixed for both devices. After signing, the NFC-enabled TPM responds with the generated quote, *i.e.*, the Quote PCR values and their digital signature Sig_{AIK}. The mobile phone compares the received PCR values with reference values and verifies the signature using the public key of the AIK. Note that the phone also verifies the public-key certificate of the AIK that was signed by a PrivacyCA. This certificate can be transmitted over the air interface or can be installed together with the application Midlet that is used to perform the attestation with public terminals. A screen-shot of the mobile phone application is shown in Figure 4.



Now searching for devices

Start remote attestation process?

Searching...

Remote Attestation

Verify trust status...

Valid ECDSA signature

Figure 4. Screenshots of the remote-attestation procedure.

The implemented Midlet is composed of 46 Java files where 20 files are used to implement the cryptographic primitive of ECDSA. 26 files have been implemented for the certificate handling, the ISO144434 connection, and the user interface. The final executable *jar* file has a file size of 122 kB.

D. Performance

The digital-signature generation of our TPM prototype takes about 31 million clock cycles. Due to the characteristics of our scenario it is sufficient to consider only the performance of a single session. Running at 13.56 MHz this results in a running time of about 2.31 seconds to generate the signature. The verification of the signed message on the mobile phone takes 33 ms. The transmission time over the air interface has been measured using an 8-bit digital oscilloscope. The time between the first bit transmitted and the last bit received has been taken. First, we measured the anti-collision and initialization phase of the NFC protocol which needs about 22 ms in our experiments. Second, the challenge N_0 and the Quote response are transmitted. For this, we assumed a typical number of different PCR values, *i.e.*, 6 in our experiments, resulting in 160+6*160+192=1 312 bits. The transmission takes about 140 ms (using a fixed RFID/NFC data rate of 106 kbit/s). Thus, the entire attestation process can be performed within three seconds. Note that we focused on a proof-of-concept realization that provides a practical demonstration of the proposed system architecture. Instead of a hardware implementation of the protocol, we implemented all routines in software. Existing TPMs already include cryptographic services such as RSA, where much more bits would have to be transmitted in contrast to elliptic curve implementations. As a comparison, the transmission of a 1 024-bit RSA-based signature (comparable with a 160-bit ECC implementation) would need 2 192 bits and roughly 240 ms transmission time which is almost twice as high as compared to the elliptic-curve based attestation protocol. This motivated our design decision, as we desire to keep the time the user needs to touch the public terminal as short as possible.

Table I shows the results of our ECC software prototype implementations. The first four lines show the code size and memory requirements of the implemented Assembler routines. Addition and subtraction need the same amount of resources which are 70 bytes of program code. Multiplication has been implemented to support different data widths and does not unroll the instruction sequences which results in 188 bytes of code. The reduction routine needs 752 bytes of code and has been optimized for the NIST reduction for $\mathbb{F}_{p192}$. The rest of the implemented files have been implemented in C. `ECC_Param.h` stores the needed elliptic curve parameters in the program code and needs 168 bytes. `ECC_FFOps.c` implements all necessary modular operations which invoke the Assembler routines. It needs 2 342 bytes of code. `ECC_PointMul.c` implements the ECC group operation that are addition and doubling which need 2 012 bytes of code. `ECC_Utils.c` and `ECDSA.c` implement utility functions such as array copying and the main loop of the scalar multiplication. Note that the implementation can be further optimized, *e.g.*, implementing all modular operations in Assembler, combining addition and doubling to reduce code size, or minimize function calls to reduce stack allocation. In total, our implementation needs 6 318 bytes of code and about 500 bytes of RAM memory. It is therefore suitable for integration in TPM circuitry.

VI. Conclusion

In this article, we proposed a new architecture that enables NFC capabilities to TPM devices. The architecture combines an NFC front-end and existing TPM functionalities into one piece of silicon and provides two additional connections for an RFID antenna. This approach allows users to simply touch NFC-enabled TPM devices with their mobile phones to verify the configuration state of a public terminal, for instance. We implemented a Midlet for the NFC-enabled Nokia 6112 mobile phone which makes a trust decision by applying the trusted computing primitive of remote attestation. The configuration states of a terminal gets digitally signed and the user gets informed on the display. Next to the mobile phone, we implemented a proof-of-concept NFC prototype that shows the practical realizability of our architecture. We implemented ECDSA on both devices and give performance results for a trust decision. We also modified the terminal software architecture to ease measurement and analysis of trusted states.

Table I
Implementation results of our NFC-enabled TPM prototype.

File	Code [bytes]	Data [bytes]
ASM_ADD	70	0
ASM_SUB	70	0
ASM_MUL	188	0
ASM_RED	752	0
ECC_Param.h	168	0
ECC_FFOps.c	2 342	121
ECC_PointMul.c	2 012	218
ECC_Utils.c	84	50
ECDSA.c	632	98

The outcomes of our work are as follows. First, it shows that trusted computing in NFC environments can effectively help to overcome confidentiality issues before the establishment of a potential distrusted terminal session. The primitive of remote attestation supported by common TPM modules can be used to provide a trust decision to users who want to establish a connection to a public terminal. Second, the rapid growth and widespread adoption of NFC in current hand-held devices like smart phones emphasize our decision for an integration of NFC into future TPMs. There already exist infrastructures for our proposed architecture such as the public-key infrastructure according to X.509 or the integration of ECC-capabilities in the Java framework. The integration of NFC into TPMs will pave the way for a ”touch’n trust” solution in upcoming applications.

Acknowledgment

The work has been supported by the Austrian government founded projects *PIT*, grant no. 825743, and *acTvSM*, grant no. 820848.

References

[1] M. Hutter and R. Toegl, “A Trusted Patform Module for Near Field Communication,” in *Proceedings of the Fifth International Conference on Systems and Networks Communications*. IEEE Computer Society, 2010, pp. 136–141.

[2] K. Adams and O. Agesen, “A comparison of software and hardware techniques for x86 virtualization,” in *Proceedings of the 12th international conference on Architectural support for programming languages and operating systems*. San Jose, California, USA: ACM, 2006, pp. 2–13.

[3] Advanced Micro Devices, *AMD64 Virtualization: Secure Virtual Machine Architecture Reference Manual*, May 2005.

[4] American National Standards Institute (ANSI), “AMERICAN NATIONAL STANDARD X9.62-2005. Public Key Cryptography for the Financial Services Industry, The Elliptic Curve Digital Signature Algorithm (ECDSA),” 2005.

[5] P. Barrett, “Implementing the Rivest Shamir and Adleman public key encryption algorithm on a standard digital signal processor,” in *Proceedings on Advances in Cryptology—CRYPTO ’86*. London, UK: Springer-Verlag, 1986, pp. 311–323.

[6] F. Bellard, “Qemu, a fast and portable dynamic translator,” in *ATEC ’05: Proceedings of the annual conference on USENIX Annual Technical Conference*. Berkeley, CA, USA: USENIX Association, 2005, pp. 41–41.

[7] E. Brickell, J. Camenisch, and L. Chen, “Direct anonymous attestation,” in *Proceedings of the 11th ACM conference on Computer and communications security*. Washington DC, USA: ACM, 2004, pp. 132–145.

[8] G. Coker, J. Guttman, P. Loscocco, J. Sheehy, and B. Sniffen, “Attestation: Evidence and trust,” in *Lecture Notes in Computer Science*, vol. 5308/2008. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 1–18.

[9] J.-S. Coron, “Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems,” in *Cryptographic Hardware and Embedded Systems – CHES’99, First International Workshop, Worcester, MA, USA, August 12-13, 1999, Proceedings*, ser. LNCS, Ç. K. Koç and C. Paar, Eds., vol. 1717. Springer, 1999, pp. 292–302.

[10] Free Software Foundation, “GNU Grub,” 2010. Available: http://www.gnu.org/software/grub/ [accessed online June 11, 2011]

[11] S. Garriss, R. Cáceres, S. Berger, R. Sailer, L. van Doorn, and X. Zhang, “Trustworthy and personalized computing on public kiosks,” in *MobiSys ’08: Proceeding of the 6th international conference on Mobile systems, applications, and services*. New York, NY, USA: ACM, 2008, pp. 199–210.

[12] D. Grawrock, *Dynamics of a Trusted Platform: A Building Block Approach*. Intel Press, February 2009, iSBN 978-1934053171.

[13] D. Hankerson, A. J. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography*. Berlin / Heidelberg, Springer, 2004.

[14] M. Hutter, M. Joye, and Y. Sierra, “Memory-Constrained Implementations of Elliptic Curve Cryptography in Co-Z Coordinate Representation,” in *Progress in Cryptology - AFRICACRYPT 2011 Fourth International Conference on Cryptology in Africa, Dakar, Senegal, July 5-7, 2011*, ser. LNCS, A. Nitaj and D. Pointcheval, Eds., vol. 6737. Springer, 2011, pp. 170–187.

[15] D. Safford, J. Kravitz, and L. v. Doorn, “Take control of tcpa,” *Linux Journal*, vol. 2003, no. 112, p. 2, 2003. Available: http://portal.acm.org/citation.cfm?id=860399 [accessed online June 11, 2011]

[16] IEEE, “IEEE Standard 1363a-2004: IEEE Standard Specifications for Public-Key Cryptography, Amendment 1: Additional Techniques,”, September 2004. Available: http://ieeexplore.ieee.org/servlet/opac?punumber=9276 [accessed online June 11, 2011]

[17] Intel Corporation, “Trusted Boot,” 2008. Available: http://sourceforge.net/projects/tboot/ [accessed online June 11, 2011]

[18] Intel Corporation, “Intel Trusted Execution Technology Software Development Guide,” March 2011. Available: http://download.intel.com/technology/security/downloads/315168.pdf [accessed online June 11, 2011]

[19] International Organization for Standardization (ISO), “ISO/IEC 14443: Identification Cards - Contactless Integrated Circuit(s) Cards - Proximity Cards,” 2000.

[20] International Organisation for Standardization (ISO), “ISO/IEC 14888-3: Information technology – Security techniques – Digital signatures with appendix – Part 3: Discrete logarithm based mechanisms,” 2006.

[21] International Organization for Standardization (ISO), “ISO/IEC 9594-8:2008: Information technology – Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks,” 2008.

[22] T. Izu, B. Möller, and T. Takagi, “Improved Elliptic Curve Multiplication Methods Resistant against Side Channel Attacks,” in *INDOCRYPT*, ser. LNCS, A. Menezes and P. Sarkar, Eds., vol. 2551. Springer, 2002, pp. 296–313.

[23] A. Kivity, V. Kamay, D. Laor, U. Lublin, and A. Liguori, “kvm: the Linux Virtual Machine Monitor,” in *OLS2007: Proceedings of the Linux Symposium*, 2007, pp. 225–230.

[24] KVM Project, “KVM - Kernel-based Virtualization Machine,” 2006. Available: http://www.linux-kvm.org/ [accessed online June 11, 2011]

[25] Q. Li, X. Zhang, J.-P. Seifert, and H. Zhong, “Secure Mobile Payment via Trusted Computing,” in *Asia-Pacific Trusted Infrastructure Technologies – APTC, Third International Conference, October 14 - 17, 2008, Wuhan, China, Proceedings*. Hubei: IEEE, November 2008, pp. 98–112. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4683087&tag=1 [accessed online June 11, 2011]

[26] C. Fruhwirth, “New methods in hard disk encryption,” Institute for Computer Languages, Theory and Logic Group, Vienna University of Technology, Tech. Rep., 2005. Available: http://clemens.endorphin.org/publications [accessed online June 11, 2011]

[27] LVM project, “LVM2,” 2010. Available: http://sources.redhat.com/lvm2/ [accessed online June 11, 2011]

[28] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks – Revealing the Secrets of Smart Cards*. Springer, 2007, iSBN 978-0-387-30857-9. Available: http://www.dpabook.org [accessed online June 11, 2011]

[29] J. M. McCune, A. Perrig, A. Seshadri, and L. van Doorn, "Turtles all the way down: Research challenges in user-based attestation," in *Proceedings of the Workshop on Hot Topics in Security (HotSec)*, August 2007. Available: http://www.truststc.org/pubs/286.html [accessed online June 11, 2011]

[30] D. Molnar, A. Soppera, and D. Wagner, "Privacy for RFID Through Trusted Computing," in *ACM Workshop On Privacy In The Electronic Society, WPES, Alexandria, Virginia, USA, November, 2005, Proceedings*. ACM Press, November 2005, pp. 31–34. Available: http://www.cs.berkeley.edu/~dmolnar/papers/wpes05-camera.pdf [accessed online June 11, 2011]

[31] NFC Forum, "NFC Forum Type 4 Tag Operation - Technical Specification," NFC Forum, March 2007.

[32] National Institute of Standards and Technology (NIST), "FIPS-186-2: Digital Signature Standard (DSS)," January 2000, Available: http://csrc.nist.gov/publications/fips/archive/fips186-2/fips186-2.pdf [accessed online June 11, 2011]

[33] B. Parno, "Bootstrapping trust in a "trusted" platform," in *Proceedings of the 3rd conference on Hot topics in security*. San Jose, CA: USENIX Association, 2008, pp. 1–6.

[34] M. Pirker, R. Toegl, D. Hein, and P. Danner, "A PrivacyCA for anonymity and trust," in *Trust '09: Proceedings of the 2nd International Conference on Trusted Computing*, ser. LNCS, L. Chen, C. J. Mitchell, and M. Andrew, Eds., vol. 5471. Springer Berlin / Heidelberg, 2009.

[35] M. Pirker and R. Toegl, "Towards a virtual trusted platform," *Journal of Universal Computer Science*, vol. 16, no. 4, pp. 531–542, 2010.

[36] M. Pirker, R. Toegl, and M. Gissing, "Dynamic enforcement of platform integrity (a short paper)," in *Trust '10: Proceedings of the 3rd International Conference on Trust and Trustworthy Computing*, ser. LNCS, A. Acquisti, S. W. Smith, and A.-R. Sadeghi, Eds., vol. 6101. Springer Berlin / Heidelberg, 2010.

[37] M. Pirker, R. Toegl, T. Winkler, and T. Vejda, "Trusted computing for the Java™ platform," 2009. Available: http://trustedjava.sourceforge.net/ [accessed online June 11, 2011]

[38] G. J. Popek and R. P. Goldberg, "Formal requirements for virtualizable third generation architectures," *Commun. ACM*, vol. 17, no. 7, pp. 412–421, 1974.

[39] M. Prokop et al., "Grml - debian based linux live system for sysadmins / texttool-users," 2010. Available: http://grml.org/ [accessed online June 11, 2011]

[40] A.-R. Sadeghi and C. Stüble, "Property-based attestation for computing platforms: caring about properties, not mechanisms," in *NSPW*, C. Hempelmann and V. Raskin, Eds. ACM, 2004, pp. 67–77.

[41] Software in the Public Interest, Inc., "Debian gnu/linux 5.0," 2010. Available: http://www.debian.org/releases/lenny [accessed online June 11, 2011]

[42] J. Stein, "Computational Problems Associated with Racah Algebra," *Journal of Computational Physics*, pp. 397–405, 1967.

[43] C. Strachey, "Time sharing in large, fast computers," in *IFIP Congress*, 1959.

[44] Trusted Computing Group, "TCG infrastructure specifications." Available: https://www.trustedcomputinggroup.org [accessed online June 11, 2011]

[45] Trusted Computing Group, "TCG TPM specification version 1.2 revision 103," 2007. Available: https://www.trustedcomputinggroup.org [accessed online June 11, 2011]

[46] Trusted Computing Group, "TCG software stack specification, version 1.2 errata a," 2007. Available: https://www.trustedcomputinggroup.org/ [accessed online June 11, 2011]

[47] R. Toegl, "Tagging the turtle: Local attestation for kiosk computing," in *Advances in Information Security and Assurance*, ser. LNCS, J. H. Park, H.-H. Chen, M. Atiquzzaman, C. Lee, T. hoon Kim, and S.-S. Yeo, Eds., vol. 5576. Springer Berlin / Heidelberg, 2009, pp. 60–69.

[48] R. Toegl and M. Hutter, "An approach to introducing locality in remote attestation using near field communications," *The Journal of Supercomputing*, 2011. Available: http://dx.doi.org/10.1007/s11227-010-0407-1 [accessed online June 11, 2011]

[49] R. Toegl, M. Pirker, M. Gissing "acTvSM: A dynamic virtualization platform for enforcement of application integrity," *Proceedings of INTRUST 2010: The Second International Conference on Trusted Systems*, ser. LNCS, Springer Berlin / Heidelberg, 2011. In print.