Towards Trustworthy Kiosk Computing

Scott Garriss* Ramón Cáceres† Stefan Berger†
Reiner Sailer† Leendert van Doorn‡ Xiaolan Zhang†

Carnegie Mellon University* IBM T.J. Watson Research Center† AMD‡
Pittsburgh, PA, USA Hawthorne, NY, USA Austin, TX, USA

Abstract

We present a system in which a user leverages a personal mobile device to establish trust on a public computing device, or kiosk*, prior to revealing personal information to that kiosk. We have designed and implemented a protocol by which the mobile device determines the identity and integrity of the software running on the kiosk. A similar protocol simultaneously allows a kiosk owner to verify that the kiosk is running only approved software. Our system combines a number of emerging security technologies, including the Trusted Platform Module, the Integrity Measurement Architecture, and new support in x86 processors for establishing a dynamic root of trust. In ongoing work, we plan to use virtual machines to support the important case where the user wishes to run personal software on the kiosk. We are also continuing to explore several open issues we have identified surrounding trust in a kiosk scenario.*

1 Introduction

Public computing *kiosks*, such as an airline check-in terminal or a rental computer at an Internet café, have become commonplace. While kiosks enable general-purpose computing without the user carrying a bulky laptop, the user must assume that a kiosk is performing only its intended function, or more specifically, that it has not been compromised by an attacker. A compromised kiosk could harm the user by, e.g., stealing private data. Similarly, the owner of a kiosk wants to ensure that the kiosk is not used to perform malicious acts for which he may be liable.

This paper presents a system in which a user, by leveraging the capabilities of a trusted personal mobile device such as a smartphone, gains a degree of trust in a kiosk prior to using it. Trust is the expectation that a computer system will faithfully perform its intended purpose. We refer to a kiosk as trustworthy if we can verify the identity and integrity of the software running on that kiosk. We do not protect against hardware attacks but note that software attacks, such as keystroke logging, are far more common.

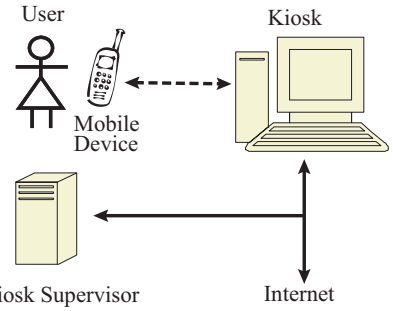

Figure 1. Kiosk computing scenario

In this context we have designed a protocol by which the mobile device establishes that the kiosk is running only trustworthy software. Only after this protocol has successfully completed will the user reveal personal information, e.g., credit-card data, to the kiosk. We have incorporated a similar protocol by which a supervisor machine, acting on behalf of the kiosk owner, verifies that the kiosk is running only approved software. If unapproved software is found, the owner can take action to disable the kiosk by, for example, removing it from the network.

We have implemented our trust establishment protocols along with the overall system to demonstrate the viability of our solution. Our prototype, depicted in Figure 1, uses a mobile phone as the personal device, a PC as the kiosk, and a BlueTooth wireless link to communicate between them. We use a second PC as the supervisor machine connected to the kiosk via the wired Internet.

The main contribution of this work to date is the experimental demonstration of a system for trustworthy kiosk computing that brings together a number of emerging se-

curity technologies and standards. We utilize new x86 processor support for establishing a dynamic root of trust on commodity computing platforms that incorporate AMD's Secure Virtual Machine Technology [5] or Intel's Trusted Execution Technology [9]. In addition, we leverage the Trusted Platform Module (TPM) [8] together with the Integrity Measurement Architecture (IMA) [14] to provide both user and owner with proof that only trustworthy software has been loaded on the kiosk. In ongoing work, we plan to use virtual machine technology to allow the user to run personal software on the kiosk in addition to software provided by the kiosk owner.

We have identified a number of open problems in the course of this work. For example, a kiosk could reboot and run malicious software after presenting proof that it is running trustworthy software but before the user reveals personal data. We are pursuing solutions to these problems and discuss our status in Section 5.

2 System Design

2.1 Technological Foundations

Trusted Platform Module (TPM): The TPM [8] is a hardware component that is increasingly available in personal computers and servers at a fraction of the cost of other secure hardware such as a coprocessor. It provides a variety of security functions, including cryptographic primitives such as signatures and secure storage for small amounts of data such as keys. The TPM is resistant to software attacks because it is implemented in hardware and presents a carefully designed interface.

Especially notable is the TPM's ability to store cryptographic hashes, or *measurements*, of loaded software components in a set of Platform Configuration Registers (PCRs). PCRs are initialized at boot time and may not be otherwise reset, with one important exception described below. They may only be modified via the *extend* operation, which takes an input value, appends it to the existing value of the PCR, and stores the SHA1 hash of the result back in the PCR. The cryptographic properties of this operation state that it is infeasible to reach the same PCR state through different sequences of inputs.

Integrity Measurement Architecture (IMA): The TPM may be used to achieve *Trusted Boot*, where measurements stored in PCRs are used to verify that the loaded BIOS, Boot Loader, and OS kernel meet expectations. IMA [14] extends Trusted Boot by additionally measuring applications and configuration files. IMA maintains an in-kernel *measurement list* containing a text description and the corresponding hash value of each software component measured.

IMA further provides an *attestation protocol* that allows a remote IMA *verifier* to challenge the integrity of an IMA platform. The IMA *attestation server* replies with the current measurement list, along with a *quote* containing the current PCR values signed by the TPM. The remote verifier then uses the measurement list to replay the sequence of PCR extend operations and verify that the resulting final PCR value agrees with the signed quote. Finally, the verifier compares the measurement list to a *measurement database* of known software, thus verifying the identity and integrity of software on the challenged system.

Dynamic Root of Trust for Measurement (DRTM): As mentioned, general PCRs are initialized at boot time and cannot be reset. Trusted Boot uses these PCRs to establish a *static* root of trust, which must include all software loaded since boot, starting with the BIOS. Recent extensions to the x86 architecture support the establishment of a *dynamic* root of trust by allowing a special PCR (PCR 17) to be reset at any time by a special CPU instruction, `skinit` in AMD processors and `senter` in Intel processors. This atomic instruction takes as input a 64KB section of code known as the *secure loader*, resets PCR 17, measures the secure loader, extends PCR 17 with this measurement, and transfers control of the processor to the secure loader.

2.2 Challenges in Verifying Software Integrity

We equip each kiosk with an IMA attestation server, and each mobile device and kiosk supervisor with an IMA verifier. To determine software integrity, the IMA verifier must have access to the expected hash values of all software components loaded on a kiosk, in order to compare them against the current values from the kiosk. It would be impractical to track every potentially relevant software component in the world so as to include its hash in a measurement database.

However, several aspects of our system combine to greatly mitigate this problem. First, kiosks often have restricted functionality, which reduces the number of software components that must be tracked, especially applications. Second, establishing a DRTM after the BIOS has run eliminates the BIOS from the Trusted Computing Base (TCB), which implies that the IMA verifier need not track numerous BIOS versions across different kiosks. Third, the mobile device must rely on a trusted third party (e.g., the user's bank or other service provider) to create the measurement database, which must be digitally signed by the third party to guarantee its integrity. The mobile device can thus obtain signed databases only as needed, either directly from the third party, e.g., via a cellular network, or from the kiosk itself.

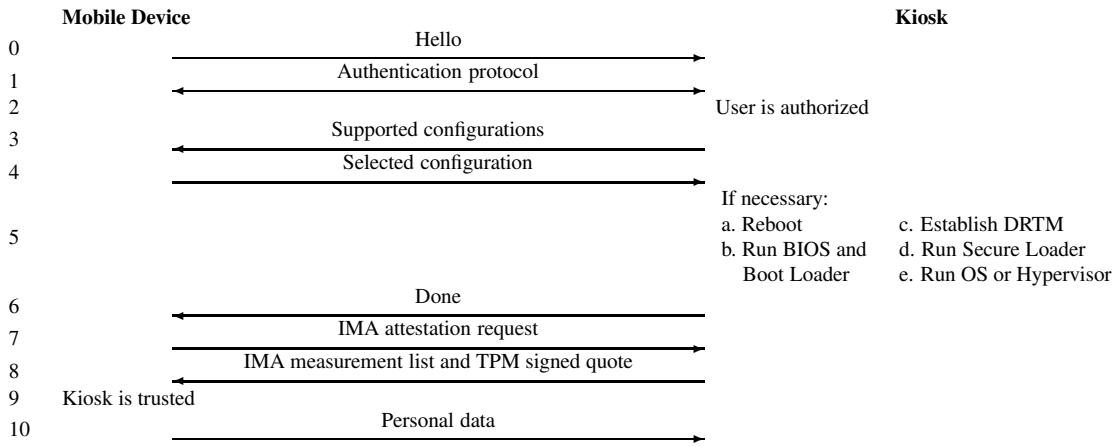

Figure 2. Trust Establishment Protocol between Mobile Device and Kiosk

2.3 Trust Establishment Protocol

Our trust establishment protocol is shown in Figure 2. The mobile device initiates the protocol (Step 0), then presents authorization to use the kiosk (Step 1). Prior to initiating the protocol, the user must have obtained this authorization from the kiosk owner. The authorization protocol is opaque in the diagram to indicate that a variety of mechanisms may be used (e.g., anonymous proof of payment), as long as they do not require the user to reveal personal data because the kiosk is not yet trusted. In cases involving free public kiosks, the authorization protocol may be omitted.

In Steps 3 and 4, the kiosk proposes a list of available software configurations, and the mobile device selects the one desired by the user. If the kiosk is not already running the desired configuration, it must reboot following the sequence in Step 5. In either case, the kiosk alerts the mobile device when the desired configuration is running (Step 6).

The mobile device then challenges the kiosk using the IMA attestation protocol (Step 7), receiving in response an IMA measurement list and signed TPM quote (Step 8). If the attestation protocol completes successfully, the mobile device decides the kiosk is trustworthy and informs the user of this fact (Step 9). The user may then proceed to use the kiosk, which may involve revealing personal data (Step 10).

3 Prototype Implementation

Our prototype comprises three parties shown in Figure 1: a mobile device, a kiosk, and a kiosk supervisor. Our mobile device is a Nokia N70 smartphone with GSM/GPRS and BlueTooth wireless connectivity. The smartphone runs the Symbian Series 60 platform, which supports Java 2 Mobile Edition (J2ME). Our kiosk is a desktop PC equipped with an AMD SVM-capable processor, an Infineon TPM 1.2, and an Iogear USB BlueTooth adapter. The kiosk runs the Xen hypervisor managed by a virtual machine running Linux. Our kiosk supervisor is a generic Linux PC.

The rest of this section describes the software we added to the mobile device and kiosk to carry out the trust establishment protocol shown in Figure 2. The kiosk supervisor simply runs an existing IMA verifier [14] to periodically request an integrity attestation from the kiosk.

3.1 Mobile device software

For the phone we wrote a J2ME application that interacts with the user, as well as a new IMA verifier for the J2ME environment that talks to the kiosk over BlueTooth. We used the Bouncy Castle [13] library for all cryptographic operations carried out by the IMA verifier, such as replaying PCR extend operations and verifying TPM signatures. Finally, we pre-loaded measurements of all the software expected to run on our prototype kiosk, including Xen, Linux, and the additional components described below.

3.2 Kiosk software

We added three software components to the kiosk platform: a new kiosk front-end application, an existing IMA attestation server [14], and a modified version of the OSLO secure loader [10].

Kiosk front-end: The front-end interacts with the phone over BlueTooth to establish the desired software configuration, reboots the machine into this configuration if necessary, and provides a conduit for the mobile phone to retrieve

measurements from the IMA attestation server. The front-end application is written in Java 2 Standard Edition (J2SE), with some help from Perl scripts to manipulate the configuration of the GRUB boot loader.

Secure Loader: Step 5 in Figure 2 outlines the kiosk boot sequence. After rebooting, the BIOS runs the GRUB boot loader, which in turn launches the OSLO secure loader. We use GRUB to load multiboot modules from disk, allowing OSLO to remain simple. OSLO establishes a dynamic root of trust for measurement (DRTM) by invoking skinit, then measures and runs the Xen hypervisor and the Linux kernel. As described in Section 2.1, skinit atomically measures the secure loader itself, stores the result in the TPM, and transfers execution to that loader.

Standard OSLO does not keep a list of the measurements done by skinit and by OSLO itself. As described in Section 2.1, such a list is needed by the IMA verifier to replay the measurement sequence. We extended OSLO to record the measurements of itself, the hypervisor, and the kernel in the Advanced Configuration and Power Interface (ACPI) table maintained in system memory by the BIOS. We used the ACPI table to communicate this list to IMA because there is no higher-level communication facility such as a file system available when OSLO runs.

There is an important subtlety in the use of a DRTM: software that runs after the DRTM is established must never invoke code that has not already been measured into the TPM. In the case of our prototype, nothing may invoke the BIOS after OSLO runs because OSLO does not measure the BIOS. We satisfy this requirement because neither Xen or the version of Linux used in Xen virtual machines ever call back into the BIOS or permit applications running inside a VM to do so.

4 Personalized Computing Environments

In ongoing work we plan to use virtual machine (VM) technology to support the important case where the user wishes to run personal software on the kiosk, in addition to software provided by the kiosk owner. Internet Suspend/Resume [11] and SoulPad [4] have shown how VMs can be used to run complete personal computing environments on kiosks. However, both efforts left unresolved the trust issues that are the focus of this work.

We plan to use the trust establishment protocols presented earlier to verify that a kiosk is running a trustworthy hypervisor environment before a user allows a personal VM to run on the kiosk. This approach allows a user to verify that the kiosk's hypervisor offers his VM strong isolation guarantees, while also allowing the owner to verify that the kiosk's hypervisor is configured to contain any VM that should engage in malicious activity.

5 Open Problems and Possible Solutions

Run-Time Attestation: The technologies described earlier guarantee the state of all kiosk software at the time it is loaded. Such guarantees represent a significant improvement over no guarantees, but these technologies do not track the state of software while it is running. Providing stronger run-time guarantees is a difficult problem and the focus of active research (e.g., [16]). Future run-time attestation solutions should be incorporated into our trust-establishment protocol, but we do not address this problem further.

Kiosk-in-the-Middle Attack: The kiosk in front of the user, or local kiosk, could run malicious software that relays data between the mobile device and a second, remote kiosk. The remote kiosk could run and attest to the intended software stack, thus fooling the device into trusting the local kiosk. The local kiosk could then snoop on and misuse personal data. This problem is similar to the chess grandmaster problem [1]. To prevent this attack we need to ensure that the TPM quote was signed by a TPM inside the local kiosk.

One possible solution involves cryptographically binding the certificate for a secure channel endpoint (e.g., SSL or IPSec) to the signing key of a TPM located on the same system [6]. If we establish a secure channel between the mobile device and a kiosk, the problem reduces to verifying that the secure channel terminates in the local kiosk. We are investigating methods for displaying a random string on a terminal endpoint that is running a known software stack in a manner that cannot be replicated on another machine.

A second possible solution involves encoding in a barcode the the public signing key of the TPM inside the kiosk. This barcode could be displayed on the outside of a kiosk using a tamper-evident medium (such as glass) and scanned with the camera present on modern mobile devices [12, 15]. Assuming no hardware attacks, if the scanned key can be used to successfully verify the signature on the integrity attestation, the user would know that the TPM doing the attestation resides inside the local kiosk.

Reboot-between-Attestations Attack: A kiosk could reboot and run malicious software after attesting to its software integrity but before the user reveals personal data. The malicious software could then misuse the personal data. Even if the user or owner repeats the attestation request, a time window would remain during which the kiosk could reboot into and out of malicious software without being detected. To solve this problem we need to ensure that the kiosk has not rebooted between the time of attestation and the time of use.

We have submitted to the Trusted Computing Group several extensions to current TPM standards that would enable remote parties to detect reboots between attestations [7].

One possibility is to incorporate a reboot counter in the TPM quote command. This problem is not specific to the kiosk scenario, and must be resolved to enable the safe use of remote attestation in general.

6 Related Work

Surie et al. [18] investigate means of establishing trust in a kiosk through the use of a simple low-cost storage device (e.g., a flash drive). The kiosk boots from this device, which contains a minimal software root of trust that uses IMA to measure subsequent software modules. Their approach emphasizes low cost but is susceptible to an attack where the software root of trust is booted within a rogue virtual machine. We prevent this attack by using a hardware root of trust, i.e., the TPM.

Asokan et al. [2] describe how a trusted server may assist a user in authenticating a kiosk. However, they do not verify the integrity of the kiosk software. Brands et al. [3] bound the physical distance between two communicating parties using time delays. Although their approach is not accurate enough to detect an attacker who is in close proximity to either party, it may help make kiosk-in-the-middle attacks more difficult. Stajano and Anderson [17] propose the use of physical contacts to establish a shared secret that subsequently enables secure wireless communication. While less prone to tampering than barcodes, this technology requires hardware not widely available on mobile devices.

7 Conclusions

We have presented the design and implementation of a system that allows a mobile user to gain a degree of trust in the software running on a public computing kiosk. The system also allows kiosk owners to enforce the integrity of kiosk software. Our prototype brings together several emerging trusted computing technologies and standards. We are continuing to explore issues raised by this work, and to propose our solutions to standards bodies where relevant. We feel that our system is making significant progress towards protecting the interests of both users and owners of public computing facilities.

References

[1] A. Alkassar, A.-R. Sadeghi, and C. Stüble. Secure object identification - or: Solving the chess grandmaster problem. In *Proc. of New Security Paradigm Workshow (NSPW)*, 2003.

[2] N. Asokan, H. Debar, M. Steiner, and M. Waidner. Authenticating Public Terminals. *CompNet*, 31(8), 1999.

[3] S. Brands and D. Chaum. Distance-bounding protocols (extended abstract). In *Theory and Application of Cryptographic Techniques*, pages 344–359, 1993.

[4] R. Cáceres, C. Carter, C. Narayanaswami, and M. T. Raghunath. Reincarnating PCs with Portable SoulPads. In *Proc. of ACM/USENIX Conference on Mobile Computing Systems, Applications, and Services*, 2005.

[5] Advanced Micro Devices. Secure Virtual Machine Technology. http://www.amd.com/.

[6] K. Goldman, R. Perez, and R. Sailer. Linking Remote Attestation to Secure Tunnel Endpoints. In *Proc. of 1st ACM Workshop on Scalable Trusted Computing*, 2006.

[7] K. Goldman and R. Sailer. Making reboot between TPM attestations visible. TCG standards discussions, 2006.

[8] Trusted Computing Group. Trusted Platform Module. https://www.trustedcomputinggroup.org/.

[9] Intel. Trusted Execution Technology. http://www.intel.com/technology/security/.

[10] B. Kauer. OSLO - The Open Secure LOader. http://os.inf.tu-dresden.de/ kauer/oslo/.

[11] M. Kozuch and M. Satyanarayanan. Internet Suspend/Resume. In *Proc. of IEEE Workshop on Mobile Computing Systems and Applications*, 2002.

[12] J. M. McCune, A. Perrig, and M. K. Reiter. Seeing is Believing: Using Camera Phones for Human-Verifiable Authentication. In *Proc. of IEEE Symposium on Security and Privacy*, 2005.

[13] Legion of the Bouncy Castle. Bouncy Castle Lightweight Cryptography API. http://www.bouncycastle.org/.

[14] R. Sailer, X. Zhang, T. Jaeger, and L. van Doorn. Design and implementation of a TCG-based integrity measurement architecture. In *Proc. of USENIX Security Symposium*, 2004.

[15] N. Saxena, J.-E. Ekberg, K. Kostiainen, and N. Asokan. Secure Device Pairing Based on a Visual Channel (Extended Abstract). In *Proc. of IEEE Symposium on Security and Privacy*, 2006.

[16] A. Seshadri, M. Luk, E. Shi, A. Perrig, L. van Doorn, and P. Khosla. Pioneer: Verifying Integrity and Guaranteeing Execution of Code on Legacy Platforms. In *Proc. of ACM Symposium on Operating Systems Principles*, 2005.

[17] F. Stajano and R. Anderson. The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks. In *Security Protocols Workshop*, 1999.

[18] A. Surie, A. Perrig, M. Satyanarayanan, and D. Farber. Rapid Trust Establishment for Transient Use of Unmanaged Hardware. In *Technical Report CMU-CS-06-176*, 2006.