ECE750-6: Pervasive Computing
Research Survey

Survey of Trusted Computing Technologies and Challenges

Michael Jarrett (99318764)

August 10, 2004

1 Introduction

Consumer computing devices from a security standpoint, can generally be classified as an 'open' system or a 'closed' system.

Closed computing devices are designed to only run specific software. Normally, the software is fixed, or if programmable, must be authenticated to be from a known source. Examples of such devices are mobile phones and modern video game consoles. Security becomes much simpler on such devices, since software can run knowing that the platform will behave as expected, and the platform is assured that the software will not attempt to circumvent system security. With the addition of protected cryptographic keys, users on the network can be assured that authenticated devices will operate in the way they expect.

Open computing devices, by design, lack any limitations on what runs on them. These systems often allow for mixing and matching of different vendors' software at a variety of levels, and almost infinite flexibility to modify any of these layers. The prime example of such devices are personal computers. However, this creates a difficulty from a security standpoint, since any one piece of software cannot trust any other piece of software, either on the local machine or over a network, to behave as expected. In fact, software in such systems must deal with security under the assumption that all other software is and even hardware may be maliciously acting against it.

It would be convenient to those using open devices and presenting services on networks with such devices if one could have the flexibility of an open computing platform, and the security and trust of a closed computing platform. Recent research has worked towards a solution, referred to as *trusted computing*[1] (TC). TC systems, through the use of cryptographic hardware and secured software, can enable a variety of security applications.

In this survey, we will discuss several of the issues and challenges in TC. Section 2 presents the four key capabilities that trusted computing initiatives wish to provide. Section 3 will discuss the moral and political aspects of TC. Section 4 will describe several real-world applications for TC. Section 5 will mention some of the efforts in industry and research to create TC systems. Finally, section 6 will mention other interesting problems and challenges TC faces.

[1] A variety of other names are often used, including trustworthy computing, and even treacherous computing. We will use the abbreviation TC, and leave the choice to the reader.

2 Foundations

TC covers a concept more than a particular technology. However, it is generally recognized that four[1] technologies will be required.

Secure I/O requires the path between user input/output devices and a running program to be protected from eavesdropping and tampering, normally through the use of encryption. This prevents attackers from compromising security though the user by manipulating what the user perceives or enacts.

Process isolation is a stronger form of memory protection, ensuring that one program does not write to the memory space of another except through controlled means of input. This is stronger than the protections currently provided by virtual memory architectures, as it even protects the process from the vast majority of the operating system, in case the OS or its components misbehave. This is often referred to as *memory curtaining*, in reference to a particular technique for implementing process isolation.

Sealed storage allows programs to store data in such a way that only the same program (in the same state) can recover it. This allows data to be protected on an insecure storage medium like a hard disk, while preventing its access from other programs, or even a modified version of the same program.

Attestation, perhaps the most powerful and controversial part of TC, allows for the hardware to sign a certificate with evidence of the state of the software at each layer between the hardware and a requesting program. This certificate can be used to prove to remote systems that the software is unmodified and running on a secure platform.

Most TC designs rely on a secure hardware module, with functionality similar to a smartcard. A private key is held exclusively within the hardware module, a public key exported from it, and this module can do cryptographic operations internally. Chipset and processor features are normally required to provide secure I/O and process isolation. Significant software enhancements are required to create an environment that uses TC, but many proposals do not require that existing applications be redesigned to execute in a TC environment.

A great deal of work on TC derives on early work on *secure bootstrap*, a technology that authenticates every step of platform startup, denying execution to programs not authorized to run. However, such a system can not be possibly be classified as 'open', so is not interesting for TC. However, authenticated bootstrap, where each component in the boot process is verified to determine access to additional secrets, is a relevant topic implicit in trusted computing techniques.

Another technology often mentioned alongside that of trusted computing is that of smartcards. Smartcards, while technically similar to the hardware support in TC, are designed to identify a user rather than a particular machine.

Yet another predecesor to TC is that of the secure coprocessor. These devices execute authenticated code directly on-chip, and provides tamper-resistance to ensure the operation of this code. These tend to be fairly weak computationally, providing few of the benefits of open platforms. TC allows one to create a *virtual secure coprocessor*, which gives similar security properties as their physical counterparts.

3 Trusted Computing Implications

A hindrance to the current efforts to advance knowledge in TC, rather than a technical challenge, is that of TC's opponents. There has been several very vocal and well-publicized objections to TC based on the fact that it can be used in several ways unfavourable to consumers. Those supporting it argue that these issues can be avoided, while still providing the advanced security features of a trusted platform.

3.1 Detractors

The largest objections to TC come from the potential uses of the technology. Digital Rights Management (DRM) is now tightly associated with the concept of trusted computing, which scares many who oppose DRM efforts. It is also possible for software vendors to use such systems to prevent interaction with competitors' software, which in an already heavily Microsoft-dominated world, is considered a serious risk.

Not surprisingly, *Richard Stallman*, the vocal leader of the Free Software Foundation, has raised significant objections[2] to the idea of trusted computing, referring to it instead as *treacherous computing*. Stallman argues that the primary uses of treacherous computing would be to limit the sharing of information, blocking alternative software, censorship, and generally removing freedoms granted users of an open platform. He argues that while such issues already exist today, TC would force it on consumers, and combined with powerful laws in the United States for the protection of copyright, could mean the end of the freedoms of open platforms.

Taken much more seriously[2] is a world-famous security researcher from the University of Cambridge:

[2]While Richard Stallman's many accomplishments both technically and politically are legendary, his tendency for obsessive ranting and stubborn idealism means many don't take his opinions seriously.

Ross Anderson. In a 2002 article[3], he describes several of the uses of trusted computing, and which industries gain value from such systems. Key issues raised were the ability of trusted computing to raise the cost of switching between competing products, the ability (both good and bad) to restrict and control content, and for illegal competition. An actively-maintained FAQ[4] addresses many of these issues and more, and argues that trusted computing will not solve many of the issues it is supposed to solve.

Several websites[5][6] have formed to oppose industry efforts based on the Trusted Computing Platform Alliance (TCPA). These sites have seen little activity since the TCPA was re-organized into the Trusted Computing Group.

3.2 Responses to Detractors

IBM researcher David Safford responds[7] to several criticisms against the Trusted Computing Platform Alliance efforts to design a standard hardware architecture for trusted computing. Essentially, he differentiates between the base technologies and bad ways it can be used, stating that people, "improperly lump together TCPA, Palladium, and DRM." Using several early IBM systems as an example, he attempts to counter arguments that such systems couldn't be disabled, or would lock out alternative operating systems. He also points out the fact that current industry designs are not effective at preventing hardware attacks by the owner to attempt to prove that limiting a user's control is not the primary objective. But most importantly, he argues against the idea that a technology should be shunned simply because it can be used as a means to negative ends.

An article[8] from Rob Enderle, who sits on the Advisory Committee of the Trusted Computing Group (TCG), insists that the technology will solve very real security problems, and that the diverse group of corporations in the TCG will prevent any single interest from taking unfair advantage from the standard. For example, it's unlikely that Microsoft could guide the technology towards blocking alternative operating systems while IBM and Sun are members.

4 Uses of Trusted Computing

TC will never gain popular support without useful applications. Outlined below are a few of the popular uses envisioned for TC systems.

4.1 Digital Rights Management

The primary use of trustworthy computing is digital rights management (DRM), a technology pushed heavily by the content industry to allow for content to be decrypted and used only in accordance with restrictions specified by the content generator. This technology can be used for a wide variety of uses, from preventing the copying of a downloaded music file, to enforcing forwarding or copying restrictions on emails, and preventing leaks of confidential data.

Trusted computing can aid DRM by securely storing decryption keys for data, and ensuring that only trusted programs are given these keys. Sealed storage is used to store the media, and such media is only provided to applications which can provide an attestation to being unmodified and approved for download of DRM-protected media. Secure I/O and process isolation ensure that other parts of the system cannot access the data while being used by the program.

4.2 Internet Peer-to-Peer Computing

Since the advent of distributed computing on the Internet, there has been motivation to cheat the system. In peer-to-peer file sharing networks, a method has already been patented[9] to prevent sharing by offering fake downloads for popular songs, and other methods include purposely downloading legitimate files slowly to prevent others from taking the place of the downloader. In distributed computations, such as the popular SETI@home project, there has already been several accusations of 'cheating', and significant workloads are duplicated to ensure that no one cheater can return incorrect results.

Trusted computing can prevent cheating by ensuring only approved clients participate in the network, and prevent data being used in nodes from being manipulated or accessed by other programs. This could even be used to protect P2P piracy networks from attack; one paper[10] describes using remote attestation and process isolation to prevent several forms of attack against P2P networks, ironically supporting the problem advocates of DRM hope TC will solve.

4.3 Software Tamper Resistance

A key security threat to software is the fact that it is trivial for it to be modified for malicious purposes. A common solution already used for many types of software is to digitally sign an application with the development team's signature key. However, this solution can be circumvented simply by modifying the set of accepted signatures, or even by modifying the underlying operating system. This is very common in rootkits, a form of exploit which modifies a running operating

system to provide backdoors while hiding its own presence from other programs in the system.

TC offers many potential solutions to such problems, including using hardware-protected keys to verify software. Even if the software is allowed to execute, applications run in a corrupted environment would not be able to get access to secure storage or succeed in obtaining resources using remote attestion. This would severely limit the damage such software could cause.

The Enforcer[11] project implements file protection on a Linux system. A Linux Security Module monitors all file accesses, and should any of those files have changed from a recorded hash, several actions can be taken, including logging, denying access, or locking cryptographic resources. The module provides an encrypted filesystem, protected by keys on a Trusted Platform Module (see section 5.1). Should any software between the BIOS and the Enforcer module be modified, the Enforcer module will be unable to unlock the filesystem.

4.4 Other Applications

A paper[12] outlines several applications of a virtual-machine based trusted computing environment. One suggestion is a distributed firewall; instead of forcing a centrally-deployed firewall, each device implements a firewall locally. A trusted module is used as a gateway to an IPSec-protected network, and can only connect using keys negotiated with remote attestation. While arbitrary software can still run on the system, no communication with the IPSec network can be performed without the firewall module running in an unmodified state.

The same system could be used to prevent spam. A common approach proposed in research to counter email spam is to force senders to solve a computational puzzle before email is accepted, however this suffers from a wide diversity in computational power of devices that wish to send email. A trusted module able to relay email could assure that email has been appropriately rate-limited, and that headers have not been forged.

A technical report[13] from the Enforcer/Bear project describes a method for using currently-available trusted computing hardware modules to create a trusted webserver. A security administrator signs particular system configurations, enforced by the trusted Enforcer program. Should the signed conditions be violated, or any tampering be detected in the enforcement part of the system, the webserver is denied access to it's content, and more importantly, its SSL private key.

5 Trusted Computing Technologies and Approaches

5.1 Trusted Computing Group

Undoubtedly a leader in the area of TC is the *Trusted Computing Group* (TCG). This group formed in 2003, as a reorganization of the disbanded Trusted Computing Platform Alliance (TCPA), retaining the design work that group had done to date. It is an industry alliance with the goals of designing a standard hardware module and software interface layer to support trusted computing applications. [14]

TCG has created two key specifications, covering hardware and low-level software respectively. The *Trusted Platform Module* (TPM) specification describes the interfaces and minimum functionality of a hardware module used for trusted computing, while the TCG Software Stack defines the behaviour and interfaces of low-level interface software between the TPM and operating systems.

The TPM is fundamentally a cryptographic processor with an embedded asymmetric keypair. The private key, called the *endorsement key* never leaves the processor, and is normally generated at the time of manufacture. This is used to generate identity keys, which are used for communication with the world. Identity keys are signed by an external certificate authority and protected by the endorsement key to ensure that the certificate actually belongs to a TPM.

Identity keys, which are certified by a certificate authority to be from a TPM, can then be used to attest to the state of software in the system. The private keys are held in the TPM, and are used to sign off on a state of platform configuration registers (PCR), which store hashes of the current system state. This is used for remote attestation by a program wishing to prove it is in a known state.

The TPM also provides for secure storage. An application may request a particular resource be encrypted using a key and the current value of a subset of the PCRs. Should the application wish to decrypt the data, the PCRs must be in the same state, which will not be true if any software changes.

The TPM 1.2 specification has been widely accepted as a standard for a hardware base for remote attestation and sealed storage. Many other initiatives assume such a module is in place. However, on its own, the TPM does not provide for process isolation or secure I/O, the other key requirements of TC

5.2 LaGrande

Intel Corporation is in the process of implementing *LaGrande*[15] technology into its chipsets. LaGrande follows up on the work of the TCG, incorporating a TPM module into its platform. It adds secure input from USB keyboards and mice, and protects the screen buffer of graphics adapters. Technology in Intel's processors will provide memory protection adequate for process isolation, as well as placing controls on the direct memory access controller through the chipset to ensure that no software (or even a hardware device) can use it to circumvent processor protections.

This technology builds upon the existing specification of the TPM 1.2 to provide hardware sufficient for the implementation of trusted computing.

5.3 Next Generation Secure Computing Base

The *Next Generation Secure Computing Base*[16] (NGSCB) is Microsoft's push to provide TC features as part of the upcoming Longhorn operating system. Formerly called Palladium, the initiative was renamed after receiving negative publicity.

NGSCB consists of a secure 'nexus', which runs parallel to the running operating system. This is a minimal operating system, providing memory management and access to cryptographic resources. On top of this run nexus compute agents (NCA), which serve as the core protected application of a trusted application. These NCAs are protected from each other and from the platform.

NGSCB also mandates a security support component (SSC), which provides hardware support for the nexus. They state that the TPM 1.2 specification is suitable for an SSC component, but would require additional chipset support for secure I/O and protected memory. Intel's LaGrande technology conveniently adds exactly this to TPM 1.2.

5.4 Other Approaches

A Microsoft paper[17] describes a layered system where each layer's software is only ever executed by layers below it. The lower layer can record the state of each process in the upper layer when it begins execution, and later use this information to determine whether or not to grant access to cryptographic resources. They describe APIs using this system to easily implement secure storage and attestation. They require a basic hardware secret plus effective memory isolation, though they state, unlike most industry researchers, that virtual memory protection can be sufficient for process isolation.

Dartmouth College's Enforcer/Bear[13] demonstrated practical secure storage and remote attestation in the context of an Apache webserver under Linux. Using a trusted bootloader and TCPA hardware, it was possible for the system to perform remote attestation simply through the use of an SSL certificate. The trusted kernel would deny access to Apache's web resources and SSL private key if the trusted root had been tampered with in any way, or if the Apache configuration changed in a way inconsistent with a configuration authorized by a trusted security agent.

Stanford has developed[12] a system based on a trusted virtual machine monitor (T-VMM). To provide strong isolation and monitoring guarantees, an unmodified OS is run on a virtual machine, while each trusted service is run on an isolated virtual machine. Minimal hardware authenticates the T-VMM, while the T-VMM can authenticate each trusted process. The use of virtual machines allows the trusted virtual machines to create devices for the untrusted virtual machine, allowing the bulk of an application to continue to run in an unprotected environment.

6 Challenges

6.1 Platform Keys

Practically all trusted computing solutions assume a keypair is permanently stored in hardware, and this key (indirectly) allows others to establish trust in the platform. However, since such keys are unique, to verify the authenticity of such a key, either the public component of such a key must be signed by the manufacturer before being stored in hardware, or the manufacturer must store the key to later authorize.

Both cases have serious shortcomings. There is no guarantee to users of the chip (since they don't generate the key) that the private portion of the key has not been retained by the manufacturer. Furthermore, it introduces privacy issues by allowing a machine to be uniquely identified, though these can be partially alleviated by the use of trusted third parties. Finally, remote attestation using such a key will require access to a public key infrastructure, which is by no means a solved problem.

6.2 Security of Trusted Applications

Despite assertions of TC supporters, most TC technologies currently do not protect software from security holes. While other applications may not be able to access secure storage, the application authorized to access the storage can still be compromised using conventional attacks such as buffer

overflows or scripting.

Additional work is needed to block an application that has been compromised from accessing the trusted environment. This will likely never provide perfect protection, but could incorporate technologies such as intrusion detection, and enhanced hardware protections from buffer overflows.

6.3 Configuration and Upgrade

If an application changes state, it should be denied access to sealed storage. However, there is often a legitimate need to upgrade or reconfigure software, which would instantly bar it from accessing protected content. The data would have to be unlocked prior to the upgrade, imported, and relocked once data import was complete.

Many systems avoid this by placing only minimal functionality within the trusted portion, however this means the untrusted portions does not benefit from verification. Ideal would be the ability for someone (preferably the owner or a party trusted by the owner) to sign upgrades, but techniques need to be developed to do so securely.

6.4 Backup and Recovery

Backup is made more difficult in trusted computing, since only an identical configuration can read sealed data. Each application must be equipped to export data for unprotected backup, which would circumvent many of the DRM uses of the system. Even worse, data is protected by a set of keys in a fragile electronic device, designed specifically to prevent those keys from being extracted. A failure of that device would mean the instant loss of all protected data.

Methods need to be explored to backup private keys protected in hardware in a manner that does not circumvent the security of the system. Software will need to be able to use those keys to recover data, or the hardware capable of importing private keys.

References

[1] "Trusted Computing," *Wikipedia: The Free Encyclopedia*; http://en.wikipedia.org/wiki/Trusted_computing.

[2] R. Stallman, "Can you trust your computer?," *Newsforge,* 2002; http://www.newsforge.com/print.pl?sid=02/10/21/1449250.

[3] R. Anderson, "Cryptography and Competition Policy - Issues with 'Trusted Computing'," *proc. 2nd Annual Workshop on Economics and Information Security (WEIS03),* College Park, MD, 2003.

[4] R. Anderson, "'Trusted Computing' Frequently Asked Questions," *Ross Anderson's Home Page*, Aug. 2003; http://www.cl.cam.ac.uk/~rja14/tcpa-faq.html.

[5] "Anti-TCPA"; http://antitcpa.alsherok.net/phpnuke/html/.

[6] "No TCPA!"; http://www.notcpa.org/.

[7] D. Safford, "Clarifying Misinformation on TCPA," report, *IBM Research,* Oct. 2002.

[8] R. Enderle, "Trusted Computing: "Maligned by Misrepresentations and Creative Fabrications"," *Security Pipeline,* Feb. 5, 2004; http://www.securitypipeline.com/showArticle.jhtml?articleID=17602019.

[9] J.C. Hale, G.W. Manes, *Method to inhibit the identification and retrieval of proprietary media via automated search engines utilized in association with computer compatible communications network,* US Patent 6,732,180, to University of Tulsa, Patent and Trademark Office, 2004.

[10] S.E. Schechter, R.A. Greenstadt, and M.D. Smith, "Trusted Computing, Peer-To-Peer Distribution, and the Economics of Pirated Entertainment," *2nd Workshop on Economics and Information Security,* College Park, MD, May 29, 2003.

[11] O. Wild, "Enforcer Homepage," Apr. 9, 2004; http://enforcer.sourceforge.net/.

[12] T. Garfinkel, M. Rosenblum, D. Boneh, "Flexible OS Support and Applications for Trusted Computing," *9th Hot Topics in Operating Systems,* 2003.

[13] J. Marchesini, S.W. Smith, O. Wild, R. MacDonald, "Experimenting with TCPA/TCG Hardware, Or: How I Learned to Stop Worrying and Love The Bear," tech. report TR2003-476, Department of Computer Science, Dartmouth College, December 2003.

[14] "Trusted Computing Group: Home," 2004; http://www.trustedcomputinggroup.org/

[15] N. Stam, "Inside Intel's Secretive 'LaGrande' Project," LaGrande Overview and Technical Insights, *ExtremeTech*, Sept. 19, 2003; http://www.extremetech.com/article2/0,1558,1274119,00.asp.

[16] "Security Model for the Next-Generation Secure Computing Base," *Windows Platform Design Notes* white paper, Microsoft Corporation, 2003.

[17] P. England and M. Peinado, "Authenticated Operation of Open Computing Devices," *proc. 7th Australasian Conference on Information Security and Privacy,* pp. 346-361, 2002.