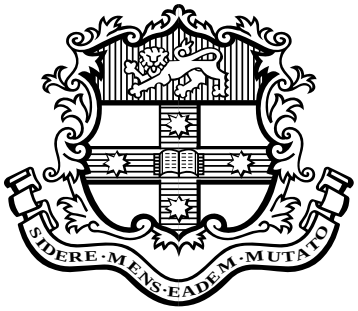
THE UNIVERSITY OF SYDNEY

UNIT FOR HISTORY AND PHILOSOPHY OF SCIENCE



The Controversy over Trusted Computing

Catherine Flick

Submitted in partial fulfilment of the requirements of the degree of Bachelor of Science Honours (History and Philosophy of Science), University of Sydney

June 2004

"But first, to reconcile the blue-ey'd maid
For her stol'n statue and her tow'r betray'd,
Warn'd by the seer, to her offended name
We rais'd and dedicate this wondrous frame,
So lofty, lest thro' your forbidden gates
It pass, and intercept our better fates:
For, once admitted there, our hopes are lost;
And Troy may then a new Palladium boast;
For so religion and the gods ordain,
That, if you violate with hands profane
Minerva's gift, your town in flames shall burn,
(Which omen, O ye gods, on Graecia turn!)
But if it climb, with your assisting hands,
The Trojan walls, and in the city stands;
Then Troy shall Argos and Mycenae burn,
And the reverse of fate on us return."

Virgil, *Aeneid* II, 19 B.C.E, trans. Dryden

Contents

List of Figures

Introduction

Modern computing is a dangerous business. Viruses, trojan horses and poorly-written software all conspire to make even simple tasks, such as reading email, risky. While some analysts believe that the correct answer to these security problems is to develop new and more security-aware applications, others have argued that these problems are endemic and cannot be fixed by anything short of a new security infrastructure. This new platform, called Trusted Computing, was recently proposed by a consortium of major tech companies. Proponents claim that it will make networked computing safe again. This thesis shows that even if it is implementable, it would only succeed at the expense of the consumer's privacy and freedom of choice.

The controversial nature of Trusted Computing became obvious to everyone, when, in May of 2004, in the middle of Microsoft's Windows Hardware Engineering Conference (WinHEC), the online technology newspaper CRN published a report that Microsoft was shelving its Trusted Computing Platform-based Next-Generation Secure Computing Base (NGSCB) security system. It quoted the product manager of Microsoft's Security and Technology Business Unit, Mario Juarez, as saying, "We're evaluating how these NGSCB capabilities should be integrated into Longhorn, but we don't know exactly how it'll be manifested. A lot of decisions have yet to be made" [Rooney, 2004]. Headlines such as "Microsoft Drops Next-Generation Security Project" quickly appeared on the popular newspaper discussion site, Slashdot [Lord, 2004], and similar headlines and articles replicated across many other news sites. Unfortunately, CRN had misinterpreted (or misrepresented) Juarez: NGSCB had not been canned, but had been slated for re-evaluation after feedback from developers and potential customers. However, speculation and rumour flooded the major Internet news sites until Juarez was urged at WinHEC to confirm whether the scrapping of NGSCB was true. He denied the claim, stating "It's absolutely not true that it's being killed." [Naraine, 2004]. However, the fact that this simple misrepresentation had been picked up so quickly and reproduced so infectiously across the news sites of the Internet, often with derisive comments about the technology, echoes the underlying discomfort felt by many Internet citizens over the plans of NGSCB and Trusted Computing. I seek to explore the reasons behind the controversial nature of this project, by examining the discussions between critics and proponents of Trusted Computing, and by drawing conclusions as to the viability of such a project in an environment where the user is accustomed

to being ultimately in control.

Proposals for solutions to the security nightmare are not new, but Trusted Computing is unique in that it is backed by the majority of large computing companies from both hardware and software realms. The controversy stems from the problems with the Trusted Computing specification, in that critics claim that the specification takes the control of the operation of the computer away from the computer user, and that it will provide an easy way for companies to force computer users into moving away from competitive software and to build Digital Rights Managements into the computing platform. In this thesis, I show that these criticisms are indeed well-founded, in that there are ethical considerations that must be made in evaluating the viability of Trusted Computing. I also explore sociological issues, to do with definition and policy, and social issues surrounding the "double-edged sword" nature of Trusted Computing and its applications.

Other sociological problems I discuss are in regard to the language of Trusted Computing. Specifications regarding computer technology open standards are usually carefully and thoroughly written, and are often open to public perusal in order to assess any potential problems before deployment. The Trusted Computing Group specification [Trusted Computing Group, 2003], although still under revision and open to public viewing, is plagued by a lack of internal consistency in its definitions, and with hardware already being manufactured that complies with this specification, this leaves little room for the specification writers to clear up such issues. I explore these and other problems with language that I have encountered in investigating the proposals for this system.

Despite the issues I examine, there are good outcomes from the research and development companies have put into Trusted Computing. The fact that these companies have decided to act on the security problems largely ignored for years is to be applauded, and as it stands, Trusted Computing could be used effectively in small domains. However, without severe modifications to the current proposals, Trusted Computing could eventually create more problems than it solves. Some suggestions for improvements have been proposed by third parties, but these may only serve to cripple the useful parts of Trusted Computing, leaving us with a similar dilemma to that we currently experience.

Overall, Trusted Computing as it currently stands is not an adequate solution to the problems faced by the average consumer, in that it would not solve these problems while maintaining the freedom of the user to control the use of their computer.

Chapter 1 focuses on a technical introduction to Trusted Computing, by explaining the reasoning behind the advances that led to its development, the current security climate, public key cryptography, the underlying requirements for and mixed definitions of trust, the hardware requirements for the Trusted Platform Module, and Microsoft's foray into Trusted Computing, NGSCB. This chapter is technical by nature, but is important as a background to Trusted Computing in order to understand the problems with it.

Chapter 2 delves into the problems associated with the Trusted Computing

architecture on a technical and sociological level. It introduces Public Key Infrastructures and the inherent problems with them, the application of these issues to Trusted Computing, as well as various ways that Trusted Computing will still be vulnerable to attack. It also addresses the issues of the "double-edged sword" nature of Trusted Computing, in which the tool-like qualities it possesses leave it open to be used for purposes that the companies involved in the pursuit of ubiquitous Trusted Computing may not necessarily wish to be associated with.

Chapter 3 discusses the viability of Trusted Computing and the controversy that has raged over it, the potential for abuse, ethical issues associated with accessibility and back-compatibility, social engineering problems, language usage, and the attempts that have been made to make Trusted Computing better for the average consumer.

An apologia on methodology

The technical exposition in Chapters 1 and 2 may seem at first to be unnecessarily internalist. I beg the patience of the reader. In Chapter 3 the sociological importance of the specifications, as expressed by the Trusted Computing Group, will begin to become apparent, and it will be clear that a broad-brush, purely externalist description of Trusted Computing would not have been an adequate starting-point.

I have included many common Trusted Computing and Information Technology terms in a glossary at the end of the paper. These terms are referenced throughout the paper with a *. I have done this in order to maintain the flow of the paper.

I have relied largely on Internet sources for both primary and secondary material. This is not because of any unwillingness to read things on paper: other research I have done has been based on more normal sources. It is because the bulk of the points I wish to raise cannot be documented without using Internet sources.

It is important to distinguish between different types of Internet source:

- those which have a known provenance, including intellectually respected authors such as Ross Anderson (whose work on Trusted Computing will presumably appear in peer-reviewed journals, but not in time for this thesis) and socially important organisations such as the Electronic Frontiers Foundation, the Trusted Computing Group, and Microsoft, which have no incentive to publish on paper at all;
- those which are effectively Internet-based newspapers, often reporting on more technical information than would ordinary print newspapers, but which should be accorded similar treatment; and
- those which merely document the (often uninformed) opinions of concerned computer users[1].

[1] In Internet terminology, often called the "unwashed masses".

I have, I believe, restricted myself to the two former categories of source, except where the cutting-edge nature of the subject means that this is not possible. As long as this important distinction is maintained, browsing the Web is not a lazy person's alternative to archive research; it *is* archive research.

Chapter 1

Trusted Computing

In order to understand the intent and methodology behind the creation of *NGSCB and Trusted Computing, it is important to understand the historical concerns in computer security that have led first to the development of the Trusted Computing base and then to the development of NGSCB. Unfortunately, space constraints prevent me from doing full justice to this history. Instead, I will briefly introduce the state that computer security is in today, with some hints about the past and with an emphasis on the issues that Trusted Computing (and NGSCB) aims to overcome. I will investigate public key cryptography, the building block of Internet-based encryption techniques, and specifically *RSA, a type of public key cryptography which is popular in many different circumstances as a way to improve security for operations that communicate over an insecure route. I will then outline the concept of *trust* in information security circles, and how it differs from some commonly held definitions and usages, and proceed to detail the hardware implementation of the Trusted Computing Platform Module, and the extensions that NGSCB add to it, in order to set the scene for some technical and sociological analysis of Trusted Computing in the following chapters.

1.1 The Current State of Security

Although it has been known since the beginning of modern networked computing history that security for computers and networks is an essential part of planning networks [Ware, 1970], underlying implementation problems that have been neglected along the way mean that today's computer users, both casual and professional, find themselves in a world of security disarray. *TCP and IP, the basic protocols of the Internet, are inherently insecure [Bellovin, 1989, Stevens, 1994]. They allow people to eavesdrop on connections, to tamper with information being sent, or to impersonate someone: for example, through "spoofing" attacks, whereby a malicious entity will pretend to be someone else by forging identification information, or through "replay" attacks, whereby entire transac-

tions are resent across the communication line by a malicious entity aiming to glean sensitive data. Other problems that currently afflict the Internet population are *viruses and *worms, *spyware, *adware, and other malicious software (*"malware"). These prey on unsuspecting computer *users and bugs in popular software, and can have disastrous effects on networks. So far, technical (as opposed to legal) approaches have primarily targeted their symptoms. Anti-virus and anti-malware software can help to keep unwanted software from causing too much damage, but these are usually only useful (or even installed) after an attack has taken place.

Software companies release *patches and upgrades to fix security vulnerabilities, but these are often not applied by administrators of the vulnerable networks and computers, and usually cannot be applied by force by software companies for fear of privacy invasion, and due to technical issues which often prevent their administration in heterogeneous networks. For example: in one recent case, a benevolent virus writer wrote an antidote worm to combat another rapidly propagating worm ("MSBlaster"). The antidote worm would scan the Internet for vulnerable computers in the same way as MSBlaster does, infect them, remove any traces of the MSBlaster worm, and download and install the security patch for the vulnerability the worm had exploited. It was widely considered even worse than the original worm because of the *bandwidth it used for downloading the security update, and because of administration issues associated with the patching of vulnerable machines [Delio, 2003]. The closest that companies can come to automatic installation of new security software is to notify the user of its existence. An example is Microsoft Windows' "Windows Update", whose frequent notifications often border on the "condescending" [Webb, 2001] and are often disabled by disgruntled users.

Yet another problem with these approaches is that it is often difficult for anti-virus and similar companies to keep up with the latest developments and post appropriate updates. With hard-hitting viruses and worms appearing more and more frequently, this is a definite concern to the health of the computers on the Internet.

It is with these issues in mind that hardware and software companies have approached the topic of an overall secure system, concentrating in most cases on addressing issues of traditional security against insecure protocols, securing the computer against becoming a soft target for determined hackers, and preventing malicious software from stealing or destroying important information.

1.2 Background Material

1.2.1 Public Key Cryptography

Trusted Computing, touted as a magic bullet for the above security issues, is built upon the well-established technique of public key cryptography.

Diffie and Hellman [Diffie and Hellman, 1976] first wrote about a system in which there is an encryption function E_k and a decryption function D_k which

work on a plaintext P such that $D_k(E_k(P)) = P$, that is, running the decryption function over the encrypted form of P results in P, the plaintext we started with. For example, if P is "This thesis is the best thing since sliced bread.", $E_k(P)$ is an unrecognisable text such as "vA56UAqkRbIP6", and $D_k(E_k(P))$ is the same as P (i.e., "This thesis is the best thing since sliced bread."). These computations assume that we have an easy and agreed way to convert text to numbers and vice versa. There are several such systems in current use, including *ASCII and *Unicode.

E_k is computed from a publically published key x, which in turn is computed from k, a private key held secret by the owner. k is required by D_k for decrypting a message, but anyone who wishes to encrypt a message to send to the private key owner can use the published key x to encrypt a message and then send it to them. The clever aspect of the Diffie-Hellman scheme is that x is chosen in such a way as to be easy to compute from k, but such that k is computationally difficult to produce from x. How this is achieved is dependent on the choice of one of many possible algorithms, but, for example, *RSA uses public keys found by multiplying two large prime numbers. The security of RSA relies on the fact that for a code-breaker to find the private key (the pair of large prime numbers) corresponding to such a public key requires finding the public key's factors, which is currently extremely difficult without a huge amount of computing power. It is worth noting that there is no reason to think that this factoring process is *inherently* difficult, and it is possible that future advances in pure mathematics will render RSA ineffective overnight.

RSA is one of the most famous and popular systems of public key cryptography [Rivest et al., 1977], and, as we will see, it is central to all current plans for Trusted Computing. To use the algorithm, keys are developed using two large prime numbers, p and q (256-bit and 258-bits respectively), and numbers d and e are sought with the property that $(de-1)$ is divisible by $(p-1)(q-1)$. The encryption function $E_k(P)$ is $E_k(P) \equiv C = P^e \bmod pq$, and the decryption function is $D_k(C)$ (where C is the encrypted plaintext that is the result of $E_k(P)$) — "vA56UAqkRbIP6" in the example above — is calculated by $C^d \bmod pq$. The mechanism works because E_k is easily calculated from (pq, e), but D_k is (currently) computationally difficult to compute from the pair (pq, e). So it follows that the key (pq, e) can be relatively safely published.

RSA is commonly used for the encryption of secure Web sessions, so that sensitive information such as credit card details, bank account details and so on can be sent over the Internet without fear of someone prying. It is also used in secure server sessions via protocols such as SSH (Secure SHell), a remote access tool (a replacement for *telnet) on most Unix-like systems. SSH uses the RSA algorithm to encrypt connection information so that passwords and other sensitive material are not sent over the Internet in plain text; instead, only their cyphered form is sent. SSH and its relatives SCP, SFTP (both file transfer protocols), SIMAP and SPOP (both email transfer protocols) provide secure alternatives to the more traditional forms of these protocols, which send unencrypted passwords to authenticate.

Even though it is currently computationally difficult to break an RSA key, ef-

forts based on distributing the workload of such an operation across a large number of computers have successfully managed a brute-force decryption [Patrizio, 2002]. As the key lengths increase, however, it takes increasingly more powerful efforts to break them in a reasonable timespan. Even so, RSA is not completely secure, not only because of the pure mathematical possibilities mentioned above but also because eventually there will probably be computing power that will be able to break the keys quickly. Therefore, those wishing to keep information encrypted for long periods of time should be aware of the fact that their data may well be vulnerable in the future.

1.2.2 Trust

In the computer and network security field the word "trust" and its cognates take on a more well-defined meaning than the everyday use of the term; and yet "trust" is still ambiguous. To disambiguate it, it is useful to define two words, "trusted" and "trustworthy" [Anderson, 2002]. Both are defined with respect to some particular security policy, thus:

- A "trusted" system is one which, if compromised, will breach the security policy.
- A "trustworthy" system is one which will not be compromised (i.e. is completely secure).

A computer that has access to sensitive information but is not secure can be described as "trusted but not trustworthy", and a computer that is completely secure (for example, not connected to a network, and locked in an underground bunker with the key thrown away), but which does not have any sensitive material on it, can be considered "trustworthy but not trusted". The ideal is to have a "trusted and trustworthy" system: a system that can store sensitive data that is totally secure [Anderson, 2002].

It is these differences between terms that caused Microsoft to change the name of its venture, of which *NGSCB is a component, from "Trusted Computing" to "Trustworthy Computing", although the Trusted Computing Group retains the former, as will I [Coursey, 2002, Anderson, 2002]. Richard Stallman, head of the Free Software Foundation, suggests that it should be renamed "Treacherous Computing", "because the plan is designed to make sure your computer will systematically disobey you" [Stallman, 2002].

The main aim of Trusted Computing is to produce a system that can be relied on to keep the information on the computer and in transit as safe as possible from unwanted outside interaction or monitoring. As we will see, what counts as "unwanted" is rather contentious; this will be addressed later. The type of cryptography used in Trusted Computing, with a unique encryption key pair for each computer, ensures that compromising the keys of one computer will not mean that other computers with Trusted Computing hardware and services are immediately able to be compromised.

However, as with almost any security system, the *users are a weak point which can be exploited through social engineering[1], a weakness that is difficult to protect against because of the requirement of users to be able to carry out their everyday tasks. The functions in Trusted Computing restrict the consequences of users being tricked into running a malicious application, etc., which would drastically reduce the problems currently experienced by such users. However, for advanced users, these mechanisms will restrict their ability to use the computer to its fullest potential. It is the lack of this distinction between types of users in Trusted Computing which leads to Stallman's accusation of disobedience, that the Trusted Computing mechanisms effectively take away any trust in the ability of the user to be able to make decisions about the running of their computer.

1.2.3 Ownership

The "owner" of a Trusted Computing platform is another ambiguous term in the *Trusted Computing Group specification. It is defined, in different places in the specification, as:

a) Any entity that knows a particular shared secret that is stored in a shielded location on the *TPM, and that may be required to prove their ownership status by producing the knowledge of this shared secret, or, if human, through asserting their physical presence to the machine, by pressing a button or otherwise.

b) The entity or person that controls the TPM, that is, the person (or human organisation) who bought and legally owns the computer. This person or their representative should be able to be verified through physical presence.

It is important to note that in some places, "physical presence" means a human being actually at the computer, while in other places it is noted that "the manufacturer of a platform determines the exact definition of physical access" [Trusted Computing Group, 2003]. The meaning of this ambiguity and the issues surrounding ownership of the Trusted Computing platform will be discussed in Chapter 3.

1.3 The Trusted Computing Platform Specification

The Trusted Computing specification is a set of rules for hardware that supplies various functions to an operating system component that works alongside it (such as *NGSCB). It aims to overcome the limitations of software security and dramatically change how computer and network security is approached.

[1] Although "social engineering" may seem like an exaggeration, it is a commonly used term in the information security community for this sort of manipulation. I take it to mean a "term [...] for cracking techniques that rely on weaknesses in [humans] rather than software; the aim is to trick people into revealing passwords or other information that compromises a target system's security."[Raymond, 1996]

The Trusted Computing specification calls for a chip, the Trusted Platform Module (*TPM), to be soldered in place next to the Central Processing Unit (*CPU) of each computer. When the computer is switched on, if the TPM is enabled, it will take measurements of the state of the *boot sequence, and record them. The *operating system that has booted is considered trusted if its boot sequence measurements are recognised as being safe. In this way, if, for example, a *virus infiltrates the system and attempts to start itself on boot, the measurements will be different and the operating system will alert the *user that it is running in an untrusted mode. When the operating system is running in trusted mode, it has access to operations that allow the user to create, encrypt, and then store files in a secure location on the hard drive, run secure programs that require the security status of the operating system to be known, and interface over a local network or the Internet to other computers that can be assured of its security status through a process of *"remote attestation". This status identification can then allow secure transmission of data to occur, with the remote computer knowing the security state of the computer to which it is sending data.

The following section details the *TPM as specified by the *Trusted Computing Group's specification, an *open standard specification that can be used by any manufacturers wishing to produce chips compliant with it, and by operating system engineers wishing to incorporate the functions the TPM provides into their operating systems. One such operating system proposed is Microsoft's Longhorn, with its *NGSCB components. Although the TPM provides many of the functions that NGSCB uses within its mechanisms, it is important to note the distinction between the two, namely, NGSCB is the software used to interface with the TPM, and the TPM is hardware. Other operating systems have TPM interaction functionality in progress, such as the open source operating system Linux.

1.3.1 Hardware Requirements

The Trusted Computing Group's specification [Trusted Computing Group, 2003] describes the requirements for such a TPM so that manufacturers can adhere to them when building compatible chips (See figure 1.1).

A communications bus allows for message transfer between the computer and the TPM. A separate component manages the information that is sent and received, and passes on the information to the appropriate areas of the TPM. It also manages authorisation and access policies. The cryptographic co-processor interfaces with hardware cryptographic engines for *RSA, *HMAC, *SHA-1, and the Random Number Generator (RNG). It also deals with encryption and decryption, key generation, and *hashing (using SHA-1), and implements a *Vernam one-time pad for authentication and transport sessions (see figure 1.2). For key generation it hooks into the appropriate component, which creates RSA keys, stores the private part in a protected area of the TPM, and returns the public key to be used.

There is an "opt-in" component within the system. This component fea-

Figure 1.1: A Trusted Computing Platform Module hardware chip

tures mechanisms that allow the TPM to be turned on or off, enabling the user to choose whether or not the TPM is to be used within their computing environment. The task of setting the status of the TPM must be authorised by the owner of the computer. (See above for how the term "owner" is ambiguous.) This can be done remotely with authorisation from the owner, but it is recommended that the owner be required to attend the machine in person (be physically present), and provide some sort of physical verification of authorisation.

1.3.2 Secure Boot

The *operating system must be *booted into a secure mode before making use of the *TPM functionality. In order to verify that no malicious additions to the hardware or software have been made, *SHA-1 hashes of measurements of configuration information are made during the boot sequence.

Each of the TPM's Platform Control Registers (*PCRs) is a 160-bit storage location for configuration information. The measurements made during the secure boot process are stored here. In order to keep the amount of information stored fairly small, a cryptographic *hashing algorithm renders a fixed-length value from the arbitrary length measurement information. Each measurement made is a hash of all the previous measurement results and the order in which these were made. In this way, the same length piece of data is always returned, although in a guaranteed greatly altered state (guaranteed in a probabilistic sense — there is always a chance that this condition might not be met, but the chance is negligibly small).

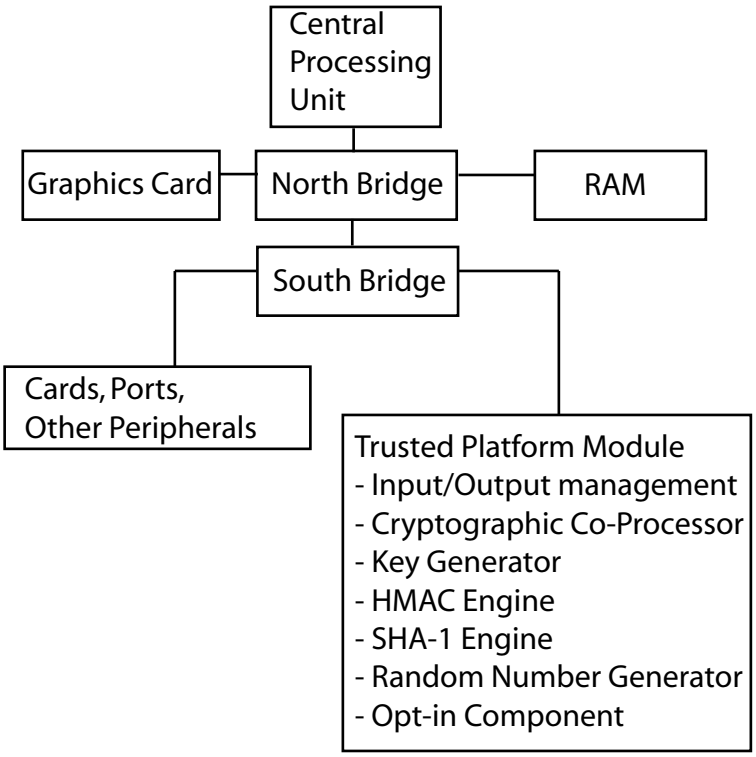


Figure 1.2: Motherboard components of a Trusted Computing Platform

This TPM-enabled boot process requires an addition to the computer's Basic Input/Output System (*BIOS), a set of instructions that runs when the computer is powered on and which loads the operating system. This addition, the Core Root of Trust for Measuring Integrity Metrics (*CRTM), is considered an initial basis for the trust in the booting system. It initially checks its own integrity, and that of the BIOS, stores this information in a PCR hash, and then passes control of the boot procedure to the BIOS. The BIOS completes its part of the boot process, takes a measurement of the state of the system, stores that in a PCR, and then passes control to the operating system loader, which makes more measurements during its boot procedure, of the operating system components and initial processes. As the operating system loads applications after booting (automatically or as required by the user), more measurements are made that store information about the state of the computer whilst running these applications in a PCR. As in the boot procedure, each of the application measurement results is a hash of all previous measurement results and the order in which these were made. The end result can be used for responding to remote attestation requests, or as an integrity signature when saving data (*"sealing") to a secure location to inform as to the security state of the computer at the time

the data was signed and sealed. If the data needs to be recovered (*"unsealed"), a request is made of the measurement hash to make sure that the state of the computer matches the security state when the data was initially sealed.

1.3.3 Use of Encryption

Users of Trusted Computers have two different types of public/private key pairs available to them. The first, the Endorsement Key (*EK), is a 2048-bit *RSA key pair of which the private portion (*PRIVEK) is stored in a shielded location inside the TPM, and the public portion (*PUBEK) is available to the user and applications on the trusted computer. This key pair is most likely to be generated by the manufacturer of the platform before it is sent out from the factory. Endorsement Keys can only be replaced following a stringent set of authentication and physical presence checks, that is, identification of "ownership" of the computer. Because there is only one Endorsement Key for, effectively, the lifetime of the computer, and because it is likely to be associated with other information about the owner of the computer, the disclosure of PUBEK could be considered to be disclosure of personally identifying information. For this reason, it is recommended that Trusted Computer *users use a second type of key pair, Attestation Identity Keys (*AIKs), as much as possible. These AIKs are also 2048-bit RSA keys which can be used by the user to certify (sign) different pieces of information about the system that remote entities may request. Since there can be many AIKs (as opposed to only one Endorsement Key), it is easier to keep each AIK separate from personally identifying information. Information that could be certified by AIKs includes information about the security state of the computer, other key information, information about the computer hardware, and so on.

For dealing with secure *sealed storage, yet another key pair is created for encrypting data that is written to storage devices. This is the Storage Root Key (*SRK), a unique key created for each owner. It is used for encryption of objects (be they files or other arbitrary pieces of data) for storage to devices such as hard drives, CDROMs, etc. Objects that are required to be used by more than one owner must be migrated accordingly, by unencrypting and reencrypting the information with the new owner's key, as objects owned by a previous owner cannot be automatically inherited by a new owner. There are currently no provisions for dealing with situations where the *TPM or motherboard break down and data must be reclaimed from the storage device — a very major oversight, since hardware failures often occur, and the data that would be securely stored on the device is most likely the most important data, thus vital to be recovered.

1.4 The Software Specification: Next-Generation Secure Computing Base

Microsoft's Next-Generation Secure Computing Base (NGSCB) is, as of writing, a combination of the hardware Trusted Computing system as defined by the

Trusted Computing Group, with a software addition, the *Nexus, a manager for software wishing to interact with the hardware functionality provided by the TPM. The Nexus provides device driver services for input/output, a file system, memory management, window management, and user debugging. The other main features it advertises are:

Strong Process Isolation: The operating system sets aside a secure chunk of memory which it uses to process internal process data with certain security requirements. The benefit of this is that if the operating system is compromised, the process table and memory can remain private and retain integrity. This is useful in protecting against viruses and some common compromise targets.

Sealed Storage: These mechanisms are related to the ability of the TPM to *seal and *unseal confidential data. Each program can store data in a secure storage area that requires authentication before it is accessible, so that the information is only accessible to the exact combination of machine, Nexus, and program that initially stored it.

Attestation: This is the process by which a program can digitally sign and acknowledge that a piece of data was created within a secure operating environment in which the software running is known and can be identified. Remote applications can use this data to identify whether the local application has integrity. This will be used to authorise carrying out of secure transfers of data across insecure protocols (such as *TCP/IP).

Secure Paths to the User: Within this framework, hardware devices (such as keyboards or mice) can be used as a secure communications route between the user and a trusted application. Keyboard input will be protected from physical attacks, and graphics card drivers will be modified so that screen shots and scrapes cannot be taken of certain areas of the screen (at least, not via the operating system — but see chapter 2), and so that physical attacks (such as *van Eck phreaking) cannot be used against the computer.

These features are presented in an accessible Application Program Interface (*API), which allows developers to write applications that use them. Microsoft has released early versions of this API to developers [Krill, 2003] so that when NGSCB is ready for release, there will be applications available that will make use of its functionality. However, the feedback on early releases of this API have made Microsoft reassess the functionality of NGSCB, an issue that is discussed further in the next chapter.

Chapter 2

Implementation and Uptake

2.1 The Problems with Public Key Infrastructure

One of the major outcomes of the Trusted Computing initiative is likely to be the advent of large-scale Public Key Infrastructures (PKI). PKI revolves around a central authoritative server, known as a "Certificate Authority" (CA), which issues electronic certificates that certify that a particular computer has a particular public key pair that identifies it. Any other computer that wishes to then communicate with this computer — and not with another computer masquerading as it — can check the authenticity of the keys sent from the computer it wishes to communicate with against the keys held by the CA (and, therefore, that it can be trusted), and then carry out communication with the knowledge that the computer is what it says it is [Ferguson and Schneier, 2003]. That this process becomes common is strongly implied by the Trusted Computing specification. Without the widespread use of PKI, large-scale remote *attestation will not be possible, and as we have seen remote attestation is a sine qua non of some aspects of Trusted Computing.

It remains unclear exactly how this infrastructure will work (if indeed it will), as privacy concerns have meant that the ability to create anonymous keys purely for communicating with remote servers has been added to the *TCG specification. These anonymous keys (the *Attestation Identity Keys) will be used for the encryption of information about the state of the computer and other non-personally identifying information, that may be requested through the direct anonymous attestation capabilities of Trusted Computing. The AIKs are certified by the *EK, but in order for them to remain anonymous, they must not be identifiable as associating with the Endorsement Key; that is, they must *not* contain information about the EK, as the EK is the unique, identifying key for that computer. If such information about the EK were to be released, it could be used to track the individual EK and thus the individual computer, invalidating anonymity. Thus the AIKs are merely a way of certifying that the

remote attesting computer that the computer is trusted, and can be used to sign information as belonging to that AIK identity, such as for communication.

2.1.1 Problems with PKI in a Trusted Computing Setting

Although Trusted Computing supports and requires PKI, there are many issues that must be dealt before large-scale PKI is feasible. These problems are general issues of PKI, but are particularly problematic in a Trusted Computing context, because of its low-level (hardware) nature and the longevity of its keys.

Expiration and Revocation

With any cryptographic effort, there is always the threat that an encryption key could be compromised. This is why many certificates and keys are distributed with an expiry date: it is easier to recover from a compromise if the keys are continually being re-generated. The idea is that those who can get hold of a certain key legitimately will be able to keep up-to-date with changes to the key more easily than those who can get hold of the key illegitimately (e.g., by wire-tapping).

If a compromise occurs, it is important for the owner of the key to be able to revoke any certificates associated with it. Difficulties that can encumber this process include:

- letting parties know that a certificate has been revoked;
- updating data that has been encrypted with the keys associated with the certificate;
- the fact that the keys are effectively locked away on the TPM and are unreadable by the user, meaning that manual revocation (i.e. without going through an application that could potentially rely on the compromised key) would be difficult for the average user; and
- the issue that generating new *Endorsement Keys (and getting them properly certified) is a particularly difficult task for non-manufacturers to undertake.

Setting up a central certificate revocation list is not an acceptable way of dealing with the problem, as it would require that all computers be online every time they need to use a key, in order to check back against the central server. Many computers are not online all the time (especially with laptops becoming more common), and so cannot easily keep track of the status of the remote certificates they use.

These issues are compounded by the fact that with the probable advent of anonymous key circulation for direct anonymous *attestation, it will be difficult to work out which keys need to be revoked without a significant privacy invasion.

Certificate Authority Compromise

The Certificate Authority that initially endorses the certificate for the main *TPM *Endorsement Key at the time of manufacture holds a lot of power, and is thus a potential target for those who wish to subvert the system. If this CA's private key is compromised, then authentic-looking TPM EK certificates could be forged. Coupled with the lack of a reasonable expiry or revocation infrastructure (as discussed above), this could be potentially disastrous for the Trusted Computing initiative. Even with multiple certificate authorities, the problem stands, as many computer owners will not understand the depth of the issues that will be at hand if they do not update their keys; not to mention the fact that updating keys and migrating data would be an immense task.

Microsoft, while agreeing that attacks on the keys stored within the TPM are possible, has claimed that "[an attack] would be an extreme case and even then would only affect the single machine (i.e., it would not be a break once, break everywhere, or "BOBE" attack)" [Microsoft, 2003a]

However, it would only require a single, targeted attack on the CA to significantly disrupt a worldwide system of remote attestation or Endorsement Key use. A targeted attack of the type that would be needed has *already* been known to work: distributed computing efforts have been successful at breaking *RSA encryption through brute-force techniques (such as with the RSA-576 key, broken in December 2003 [Laboratories, 2003] — even though this is a much smaller key than those proposed, it is theoretically and likely possible for these keys to eventually be broken similarly), and other, more traditional attempts at theft of a key are also possible, such as physically stealing the computer upon which the key resides, in order to perform physical attacks against the chip or execute physical presence authentication in order to access the computer's capabailities.

2.2 The Vulnerabilities

Although the Trusted Computing system renders its computing platform less vulnerable to some security flaws, it enhances other potential security vulnerabilities. One of these is an increased vulnerability to *Denial of Service attacks. Denial of Service (DoS) attacks involve an attacker using up the resources of a victim to effectively disable it. This can be accomplished through starving the victim's network of usable *bandwidth, filling up hard disk space, *file handle and *process table exhaustion, or any other method in which limited resources can be exhausted [Schuba, 2000]. These attacks can come from several computers at once, in a malicious orchestrated campaign. The attacking computers are often ones that have been hacked into or made vulnerable by viruses, worms, or mis-management (through not updating software, or misconfiguring software), because using such computers makes it easy to obscure the origin of the perpetrator.

DoS attacks can clearly be aimed at Trusted Computing remote attestation services — either at an attesting computer, or at companies that require remote

attestation. By flooding the bandwidth required for remote attestation, the attestation service can be stifled.

But *DoS attacks are not only used against remote computers. They can also be used against applications or hardware within one computer. The problem of protecting against this popular attack is where Trusted Computing is most vulnerable. The *TPM itself is open to attacks against crucial functions which require authorisation. Because the TPM does not keep track of previous authorisation requests [Trusted Computing Group, 2003], failed attempts at authorisation are not noted against other failed attempts; that is, the current authentication attempt is not corresponded back to any others. However, because this leaves the TPM open to brute-force dictionary attacks (that is, trying every possible combination of characters in order to eventually guess the password, which can also potentially be used to flood the *bandwidth to the TPM in a DoS as well), countermeasures against this sort of attack must be implemented. Such countermeasures can lead to the opening of the TPM to DoS attacks, crippling the TPM for other authentication requirements at the time of attack. For this reason, the TCG specification recommends implementing software-based services outside the TPM to monitor attempts at authentication, and to balance the response to such problems; but the software services that the specification calls for will themselves be open to DoS attacks. The Trusted Computing Group specification is not fixing the problem, merely failing to take responsibility for it, and will rely on software implementations to adequately protect it against DoS attacks.

Other vulnerabilities in Trusted Computing could well be discovered if the hardware could be simulated or monitored physically. IBM states it is not attempting to secure the TPM against its user, and so its chip is able to be monitored and is susceptible to power, radio, or timing analysis. "We simply are not concerned with threats based on the user attacking the chip," they claim [Safford, 2002b]. Microsoft, however, is a little more cautious with its approach, aiming at offering users a "secure pathway from the keyboard through the computer to the monitor screen, preventing it from being secretly intercepted or spied on" [Microsoft, 2003b], and considers the possibility of someone attacking the hardware as being "technically feasible" but an "extreme case" [Microsoft, 2003a], which implies that they are most certainly concerned with attacks against the chip. Whether or not Microsoft will be using IBM's chip is as yet unknown. If they do, there will be either an interesting conflict or some eating of words.

It is important to note that other security systems are also vulnerable to DoS attacks similar to those described above. Whether there is a security system as *broadly useful* as Trusted Computing which is less vulnerable to DoS attacks is a difficult question which I cannot answer (and which is perhaps too vague to answer). Whether or not it is better to have smaller, more targeted security systems that could fail under a DoS but not incapacitate the greater platform is also a subject that would require further research. However, it is safe to say that these vulnerabilities preclude Trusted Computing from claiming its magic bullet status, and that shuffling hardware problems off into software domains

will only abstract the security of Trusted Computing and make it harder to keep implementations standardised.

2.3 Ubiquity

One of the requirements for Trusted Computing to work effectively is the wide-scale uptake of the technology. In the jargon, it must be "ubiquitous", at least in some domain of computing (such as the banking domain) and preferably (from Microsoft's point of view) in all or almost all domains. Without ubiquity (at least in some domain), Trusted Computing will not have enough interoperability to be useful: in other words, without ubiquity services will fail because some component doesn't understand another (Trusted) component's messages. For example, remote attestation is useless unless many computers all trust the same attestation server.

Many other technologies have become ubiquitous in this sense, from 8-bit bytes to *ASCII to VISA card numbers to *TCP/IP. Assumptions of ubiquity must feature in early promotional material (such as [Hewlett-Packard, 2003]) so that computer hardware and software vendors become likely to establish provisions for the development of hardware, software, and policy support for the initiative, which would then inspire consumers to purchase the equipment and software. Microsoft has discovered the difficulties of needing feedback from potential clients on pre-release technology, and has also discovered the conflicting requirement for final software *APIs be available to developers early on in the release cycle in order for them to begin programming.

So far, the clients Microsoft has approached to test the technology are precisely the initial target audience Microsoft has in mind: medium-to-large businesses. In this section, I look at this target audience for ubiquitous Trusted Computing, and the difficulties associated with breaking the seeming "Catch 22" ubiquity situation in this case. I also address some issues with dealing with the everyday user, that is, the average consumer, for when Trusted Computing spreads outside the realm of specialised use.

But first I will say something *positive* about Trusted Computing.

2.3.1 Benefits to Businesses

One area in which Trusted Computing will provide much-needed services is for the networking needs of medium-to-large businesses. The secure authentication mechanisms offered by Trusted Computing greatly enhance current measures which are easily breakable through technical or *social engineering attacks. Businesses could use Trusted Computing authentication mechanisms to provide employees with secure connections to work from home, in the knowledge that the employee's computer will not have viruses or malware which could compromise the security of the connection or infect machines within the firewalled business network with viruses or worms [Anonymous, 2004]. *Digital Rights Management could be used to restrict confidential files so that the intended recipient

is the only entity able to view them. Financial transactions could be made more secure than they are currently, with financial servers able to use remote attestation to verify the identity of the computer user, thus helping to prevent success at "phishing" (a social engineering attack in which scammers pretend to be an official financial or other server and lure gullible account-holders into entering sensitive information about their accounts) and other online fraud. Another benefit of Trusted Computing is preventing employees from accidentally installing *malware and *viruses or reducing the impact of an employee accidentally giving out sensitive information such as passwords.

These benefits are important to note, as they could well be the basis for establishment of ubiquity for Trusted Computing. If Trusted Computing is successful in this arena, then it is likely that the companies selling it will shift their aim to home users, or that the rise of the platform in businesses will mean that some start to bring it home with them, thus introducing the technology to a different domain.

2.3.2 Microsoft's NGSCB Balancing Act

According to a press reports, NGSCB was canned during the 2004 *WinHEC conference. These reports were followed by Microsoft's hasty reassurance that it was "alive and kicking" after all [Naraine, 2004]. The fact that this press exchange caused such a stir in the Internet community[1] is highly reflective of the controversial nature of the project. With the next version of Microsoft's flagship operating system, Windows "Longhorn", in developer preview and with not-too-distant-future release dates tentatively declared, Microsoft had been eager to bring developers on board to write applications which take advantage of the security features of NGSCB, and had been actively pursuing clients that would want to incorporate it into their planning. However, in a somewhat short-sighted move, Microsoft ignored the potential, and now obvious, problem that most business clients would not want to rewrite the software they had been using for years (and probably commissioned for a lot of money) merely in order to use the security benefits of NGSCB. For example, Mario Juarez, the product manager of Microsoft's Security and Technology Business Unit, noted in an interview, "A major bank in the U.K. is interested in NGSCB but they have concerns about changes that they would have to make to their applications. Then there's a big manufacturing company in the U.S. that likes the idea of strongly authenticating remote employees and partners. But they want it available out of the box and not [to] require changes to applications [(programs)]." [Naraine, 2004]

That Microsoft completely missed such a serious problem in their high-profile move towards a secure operating system shows a lack in understanding (on the part of their strategic planners) of how interested parties and developers would be using the system. The fact that they had been releasing details about the *APIs to developers shows that they understood the need for a platform to have usable applications developed for it, so that customers will buy it. Of course

[1] Not to mention nearly giving the author of this thesis a heart attack!

Microsoft develops some applications for its own platforms; but it is generally thought, not least by Microsoft planners, that a single team of developers is not enough to anticipate the market's needs for a wide variety of applications. However, they misjudged the requirements customers have for such a platform, and to have moved too soon to make the APIs available to developers, many of whom have begun programming to these requirements, shows a lack of foresight in both requirements and time management. This misjudgement makes one wonder whether the hype both for and against Trusted Computing and NGSCB is at all justified; the decision to re-evaluate the system now will push the release date for it back by quite a while, if indeed NGSCB will be released in this form at all. The re-evaluation will most likely dramatically change the way that the system works, at least for existing applications, as reforming it to allow for applications to use it without need for rewriting them is a huge task. It is extremely difficult, perhaps impossible, to create APIs for NGSCB that do not require significant program adjustment without inheriting the security problems that plague current programs, rendering NGSCB useless for many security issues that it would, in its current state, fix. For example, existing calls to writing data to a hard drive would have to be modified in order for the applications to write data securely, which could seriously restrict applications that currently interoperate within Windows. The whole debacle also raises the question of whether Microsoft has adequately recognised and evaluated public opinion on the mechanisms they will be offering with this technology. Curiously, Microsoft has since "archived" all of the information on NGSCB previously directly available from its website, as if to say that the re-evaluation of NGSCB will render current information about it irrelevant [Microsoft, 2003c].

2.3.3 The Lowest Common Denominator

One of the greatest barriers to establishing the ubiquity of Trusted Computing is the problem of varying computer skill levels among users. Designing user interfaces for security services is a difficult task: even more so than other user interfaces, because of the security issues at stake. Whitten and Tygar describe a set of properties that security application user interfaces must have:
"security software is usable if [and only if] the people who are expected to use it

1. are reliably made aware of the security tasks they need to perform;
2. are able to figure out how to successfully perform those tasks;
3. don't make dangerous errors; and
4. are sufficiently comfortable with the interface to continue using it."

[Whitten and Tygar, 1998]

These guidelines are most suitable to single-task programs, and cannot easily be applied to Trusted Computing and its software handlers, because of the wide range of tasks that Trusted Computing is capable of. The software that must be written to deal with these tasks will need to cater to a wide range of users,

from new computer users through to experts who require that there be low-level interfaces for custom security settings. Having to dumb down security not only makes it irritating to these expert users, but also is very difficult to encourage users to maintain, as many users do not want to be nagged into appropriate security behaviour by the operating system.

Thus the software developers for Trusted Computing will need to balance the usability of their software and the requirements for security against a different set of imperatives from Whitten and Tygar's. They will have to make sure that the security of the system:

– is not compromised through the user's lack of interest or inability to understand the importance of security directives;

– cannot be compromised through social engineering techniques;

– is unobtrusive and allows the user to carry out their tasks without interruption; and that it

– actually provides features needed by the users of the software (such as fine-tuning for administrators).

Maintaining this fine balance is a difficult task, and requires not only intuitive software but also a large amount of education for its users. Otherwise, Trusted Computing's ubiquity will not be established and maintained.

2.4 The Law of Unintended Consequences

The overall ability for Trusted Computing to secure communication paths, sensitive data, and inter-process communication allows for many applications to reap the benefits of this sort of security. However, the ability to protect data and communications in this way is a double-edged sword, and could easily be used by organisations and people undertaking undesirable activities — illegal, "underground", or just plain *bad* — to remain anonymous and protect their sensitive information. Of course, exactly which activities fall into these categories is highly contentious; but all I need show here is that there is a problem and, since everyone agrees that *some* activities are undesirable, that is easily done. I will assume, further, that allowing illegal activities is undesirable, although I recognise that this assumption is not straightforward to interpret and may well be false.

2.4.1 Anti-Digital Rights Management

Among the major beneficiaries of the Trusted Computing intiative are those companies (and, on a smaller scale, individuals) wishing to seek to control the use of data, especially media, through *Digital Rights Management technology.

If Trusted Computing were to become sufficiently ubiquitous, the distribution of restricted-use media could be protected by requiring clients to check with

a centralised server (acting as a certificate authority) to determine the access levels for the user. This could be used to restrict people who use the media to a particular piece of software (including, potentially, a particular operating system). This is not a new problem, but incidents in the past where it has been attempted have seen it fail miserably. Notable examples are the DVD CSS (Content Scrambling System) [McCullagh, 2000] and audio encryption (including Apple Computer's "FairPlay" DRM for music downloaded from their iTunes Music Store [Fisher, 2004]).

Most of these failures have been due to dedicated parties reverse-engineering the encryption processes and finding weaknesses in them, in order to make them work with operating systems for which they weren't official software products. These parties are often strongly supported by the communities surrounding the rival software, for example, after a programmer was taken to court over the distribution of the code for descrambling the DVD CSS, other programmers pushed the code into the legal category of free speech by devising humorous and artistic ways of making the code "free speech"[2]. The original programmer's case was eventually dropped, leaving its exact status in US law extremely unclear. (It seems to me that US law embodies a self-contradiction on this issue; but I do not have space to give full details of this view.)

Other anti-DRM efforts, including efforts against FairPlay, involve circumventing the DRM process by using the key that was legally provided along with the media for the licensed player to access it, and then recording it to another, unprotected file, or by using low-level hardware (such as a sound card driver that captures the sound information being fed to it by the music player and then re-records it to an unprotected file). For example, MediaMax CD3 DRM can be circumvented by holding down the "Shift" key of the computer as it starts up! [Halderman, 2003]

In any case, even extremely simple (if somewhat old-fashioned) approaches can be taken to circumvent DRM. Let us consider this topic separately for the video domain and the audio domain:

Video: Video information can be captured from *any* computer system capable of playing the video on its own screen, no matter what security systems it has. Methods for doing this range from the simple (taking an external video of the screen) to the hi-tech (hijacking the video card to divert video output to a recorder).

Audio: Similarly, audio information can be captured from any computer system capable of playing the audio, by holding up a microphone to the speakers and recording the music to tape. Of course there are often quality degradations with this approach, but the degradation from a recording made with a good microphone positioned next to an electrostatic speaker would be inaudible, or close enough to inaudible for most purposes.

The effective outcome of the processes mentioned above, and many more like them, is that determined individuals will usually be able to crack or circumvent

[2]For a gallery of these including the code for the descrambler in t-shirt, haiku, interpretive dance, and song form, visit http://www-2.cs.cmu.edu/~dst/DeCSS/Gallery/

DRM, no matter how strong the encryption is that holds it. The number of poor quality music and video files that are currently circulating filesharing networks is testament to the fact that many people don't mind some quality degradation if the result is free. The fact that such degraded quality copies are available even without widespread, rigid DRM constraints on this type of media furthers this argument. These individual users could easily use Trusted Computing technology to create anonymous file-sharing networks to circulate code or illegal copies of media that have been freed from the DRM constraints.

2.4.2 Anti-Social and Illegal Uses

Trusted Computing offers much in its arsenal for keeping data secure from tampering and unwanted viewing by third parties. As well as being attractive to honest applications, its capabilities could be used by those wishing to keep their information secure due to the anti-social nature of that information. Whistleblowing could be prevented if incriminating documents could not be passed on to outsiders, and terrorist organisations would be more at liberty to use the Internet to perform document exchanges without fear of monitoring by international agencies.

Secure distributed efforts to crack encryption could also use the anonymous key generation capabilities of Trusted Computing to remain anonymous. *Virus and *malware writers could also use such networks to distribute information regarding the creation of such software, or for people who attempt to use such scripts to attack hosts to congregate anonymously.

In these cases, it might be reasonable to expect that international agencies would require a "back door" to Trusted Computing mechanisms so that monitoring of illegal activity could, in fact, occur. Microsoft, at least, claims it "will never voluntarily place a back door in any of its products and would fiercely resist any attempt to require back doors in products" [Microsoft, 2003a].

Whether Microsoft will defend its claims remains yet to be seen.

In this chapter, I have shown that although there might be some benefits to Trusted Computing for some applications (such as for deploying to businesses), that ubiquity in this domain will inevitably lead to ubiquity in other domains, and there are issues with dealing with everyday users that will appear as a result of this. I have also demonstrated there are some serious problems with the mechanisms being employed within the specification, including issues with deployment of Trusted Computing Public Key Infrastructures, vulnerabilities, and abstraction of security concerns from the hardware to the software implementation. I have also discussed some of the social (or anti-social) issues that could arise from a widespread deployment of Trusted Computing.

Chapter 3

Implications of Trusted Computing

The issues raised by Trusted Computing are both technical and social. The issues are interesting in their own right, and so are the ways in which the proponents and critics of Trusted Computing deal with them. So too are the social relationships between these authors. Most of the writers who see significant problems with Trusted Computing are well-respected academics and members of free speech organisations. Those defending Trusted Computing often work for the companies involved or wish to remain anonymous. This makes for an interesting sociological contrast. In the following sections I will make use of this contrast in analysing the background to the arguments for and against Trusted Computing, in addition to saying something about their validity. I will also explore issues which have not been covered adequately by the discussions to date.

3.1 Proponents and Critics

The possibilities created by Trusted Computing have aroused the admiration of many. This is evident in the number of companies that have joined the Trusted Computing Group. Most of the big name computing hardware and software companies are members [Trusted Computing Group, 2004]. Most of these have been struck in one way or another by problems relating to the insecurity of current networks and computers, and believe that Trusted Computing can give them an edge in the security arena. The all-encompassing approach of Trusted Computing has proven attractive to companies wishing to improve the overall security of their products, to the point that even those companies which disagree with the concepts have joined, purely to keep up with their competitors in the market.

Of course, developers from member companies are distinct proponents of the technology, with some going out of their way to rebut arguments against it

[Safford, 2002a] in an unofficial capacity. Other vocal advocates remain anonymous [Anonymous, 2004] or restrict their arguments to those of official statements [Hewlett-Packard, 2003, Microsoft, 2003a]. There are obvious commercial reasons why these companies stand to benefit from defending their products, even if they aren't the best solution, because of the amount of time and money that has gone into them in research and development. Whether or not Trusted Computing is the best solution to the problem of modern computer and network security, there is no reason for these companies to cast any doubt on the viability of their product, as they are not bound to deliver on any of the claims they make of their software and hardware until it is "live". Even with the positive aspects of the currently slated Trusted Computing model (as discussed in Chapter 2), any statements of its viability or benefits to computing society from the companies which stand to make money from it should be assessed sceptically.

Trusted Computing has raised the ire of academics, journalists and other writers. The issues vary from problems with implementation to civil rights issues and questions about the motives of the companies involved. Some offer solutions to the issues they investigate; others merely wish to raise awareness and educate. Although one could argue that their arguments might be seen as worst-case scenarios or "slippery slope" arguments, I would argue that the evidence, such as Microsoft's antitrust suits and ongoing anti-competitive behaviour, and Hewlett-Packard's anti-consumer activity in the realm of printer cartridges, shows that at least some of these companies have behaved similarly in the past in all-too-similar circumstances [Anderson, 2002, Anderson, 2003, Coursey, 2002, US Department of Justice, 2004]. Thus the arguments of the many critics should be taken seriously by policy makers or other interested parties wishing to understand the potential uses, abuses, and problems with Trusted Computing.

Most issues dealt with by critics are related to civil rights. The issue of user control is a problematic one, with many wondering whether Trusted Computing really will deliver more control to the user, as has been proclaimed by some advocates [Microsoft, 2003a]. In fact, they wonder whether it will instead increase the control the software and hardware manufacturers have over the user [Anderson, 2002, Anderson, 2003, Stallman, 2002, Coursey, 2002], through de facto standards set through increasing market share (as ubiquity in one domain will eventually lead to ubiquity in other domains, because the lines between these domains are often, in reality, blurred). This idea of the fine balance between trust and control is addressed by these critics in an indirect way, and I explain the real problems with this balance to a greater degree in the next section. The problems of the Trusted Computing user privacy and rights models are also hotly debated, leading to several suggestions for improving the Trusted Computing specifications. However, these may not necessarily improve the experience for Trusted Computing users at all, but are important ones and must be addressed before Trusted Computing becomes widespread.

3.2 Language

The language used by the companies involved in the Trusted Computing initiative is oftentimes ambiguous, in sharp contrast to the language more usually used in the specification of new computer systems or protocols [Information Sciences Institute, 1981]. Most of the time these ambiguities either serve to confuse or mislead, or are the result of inadequate definition to begin with. I attempt here to analyse the uses of various words throughout the Trusted Computing specification and other publications surrounding Trusted Computing, and the implications these uses may have.

3.2.1 "Trust", "Control", and the "Opt-In" Mechanism

In the Trusted Computing field, the meanings of "trust" and "control" overlap significantly with each other. Foucault [Foucault, 1975], in his famous dissertation on panopticism, introduces the concept that the two are closely related, united in the practise of *discipline*. He describes disciplinary power as being

> "... exercised through its invisibility; at the same time it imposes on those whom it subjects a principle of compulsory visibility. In discipline it is the subjects who have to be seen. Their visibility assures the hold of the power that is exercised over them."

The extent of the control that the Trusted Computer holds — being able at all times to provide a unique identity for the computer user, and responding to remote queries about that identity to disclose the activity carried out by that identity on that computer — significantly detracts from the advantages (to the user) of trust that the computer is running to a known configuration. And yet the trust could not be had without the control: they are linked in a way which I hope has become clear in Chapters 1 and 2 (and which, incidentally, could not have been seen without some perhaps internalist-seeming technical exposition). Sewell [Sewell, 2002] summarises this eloquently thus:

> "... here trust and control are not antagonists engaged in a battle for supremacy but complementary components of a disciplinary practice."

The parallels with Foucault's view of discipline can be drawn further: there are parallels between his discussion of the practice of discipline and Microsoft's vague around issues of how the NGSCB should be used. Microsoft claims that it is not dictating policy, that is, what its NGSCB should be used for, but mechanism: the functions available for software writers in developing for NGSCB [Microsoft, 2003a]. This lack of policy-making does not preclude a lack of power, however, as policy can easily be introduced in a de facto form:

> "It was a question not of treating the body, en masse, 'wholesale' as if it were an indissociable unity, but of working it 'retail', individually; of exercising upon it a subtle coercion, of obtaining holds

> upon it at the level of the mechanism itself — movements, gestures, attitudes, rapidity: an infinitessimal power over the active body." [Foucault, 1975] (See also [Grossman and Webb, 1991].)

This sequence of small movements towards a hold of power is reflected openly in the widely publicised "opt-in" nature of Trusted Computing. Giving users an option on whether to turn on the Trusted Computing mechanisms seems like a reasonable effort to keep it controlled by the user, but in reality, if Trusted Computing becomes as ubiquitous as some vendors seem to think it will [Hewlett-Packard, 2003], the very process of carrying out everyday work will require it to be on, as access to encrypted files and authentication to servers requiring attestation will otherwise be denied.

Not only will Trusted Computing be disciplining the user of the technology, it will also be disciplining the computer, separating out its programs' access to sealed storage and memory, and only allowing programs with correct authorisation to access other programs' information. This parallels the requirements of the partitioning mechanisms suggested by Foucault, in which "one must eliminate the effects of imprecise distributions, the uncontrolled disappearance of individuals, their diffuse circulation, their unusable and dangerous coagulation" [Foucault, 1975]. Partitioning the computer not only allows the programs that can be trusted to run knowing that no other program can access their data, but also allows trouble-making programs to be isolated from causing damage to other, "better behaved" programs. Thus the trusting of a computer's hardware and software by other hardware and software becomes an issue of battling control between different software authors, in such a way that two authors who do not trust each other may write this distrust into their software, preventing them from interacting or sharing each other's information where otherwise they may have. This locking down of software interoperability will affect mostly applications that attempt to mimic or work with hegemonic commercial applications such as Microsoft Office, and is likely to enhance monopolies or oligopolies, and stifle small-scale software developers, researchers and hobbyists. The control is in the hands of those who can easily discipline their users and programs to take advantage of the Trusted Computing mechanisms, and the trust gained is an imposed trust.

3.2.2 "Ownership"

Further to these issues of control, as mentioned above, the concept of "ownership" of a Trusted Computing platform within the Trusted Computing Group specification is ambiguous if not outright confused. The lack of internally consistent definitions undermines the usefulness of the concept as a whole. If the manufacturer adheres to the part of the specification which says that "the manufacturer of a platform determines the exact definition of physical access" [Trusted Computing Group, 2003], it could potentially allow programs to assert themselves as owner, taking control of "ownership" functions such as the high level administration of the TPM keys, meaning that programs could control the

TPM administration of the computer independently of the computer's owner. In this way, objects (programs) become agents in a much stronger and more insidious sense than ever intended by Latour! [Latour and Woolgar, 1986] This could impact the human owner's ability to control their own computer and, furthermore, would almost certainly place trust in the appropriate functioning of the computer with the software writers. Administration tasks would also have to be somewhat automated in order for them to be palatable to an everyday user audience, which would require the definition of "ownership" to at least allow the human owner to delegate responsibility of these tasks to a program. If such a user could be tricked into allowing a rogue application to have these priveleges, it could seriously undermine the security of the computer.

3.2.3 Name Changes and "Archiving"

When Microsoft's NGSCB first made an appearance, it was named for Palladium, the statue reported to have guarded Troy. After the initial furore over the proposed security solution established it as being overly restrictive [Anderson, 2003, Boutin, 2002, Manjoo, 2002], Microsoft changed the name of its fledgling security project to Next-Generation Secure Computing Base. Stefan Bechtold suggests, somewhat tongue-in-cheek, that this name change came about because "Microsoft was reminded that, after Odysseus and Diomedes had succeeded in stealing Palladium from the temple of Athene in Troy, the Greeks were able to capture the city some 3000 years ago." [Bechtold, 2003] Microsoft claimed it was due to a legal challenge to the rights to the name, but critics saw the name change as Microsoft reacting to bad publicity by attempting to diminish the overall effect such publicity would have on the underlying Palladium concepts [Lemos, 2003]. With Microsoft not known to bow to legal challenges on nomenclature, this explanation is very plausible. It also raises the question as to whether the "archiving" (ditching) of the NGSCB information from the Microsoft NGSCB Web site is another attempt at rebranding Microsoft's security project in order for Microsoft's marketers to try again with a clean slate.

3.2.4 "Neutrality"

Asserting neutrality over the potential uses of the Trusted Computing system is an easy way for providers to escape liability for any applications developed for it. The companies involved seem to realise the "double-edged sword" nature of Trusted Computing, that is, that it could be used for illegal and anti-social applications as well as for applications they would want associated with their company. Microsoft, in claiming its policy-neutral status, writes:

> "The NGSCB architecture offers policy enforcement mechanisms that can benefit many parties, but enforcement remains in the control of users. Whether or not they use it in conjunction with rights management systems, people can use such systems to maintain confidentiality, to control the sharing of their documents, to collaborate

> online with friends, co-workers or colleagues, or to control sensitive operations performed on their machines." [Microsoft, 2003a]

In this way it delegates all responsibility to the applications that take advantage of NGSCB. This is an easy way to evade criticism for any de facto policies that would arise, such as the use of particular pieces of software that restrict interoperation due to sofware lock-in (as discussed above). Despite claims of neutrality, Microsoft is in a uniquely positioned as a member of the Trusted Computing Group and the maker of the world's most popular operating system and software. Microsoft holds at least two types of market power:

- it can use its ability to access NGSCB *APIs early to deliver software ready for NGSCB before competing third parties can (with the possible exception of other members of the Trusted Computing Group); and
- it can use its existing market share (through current licensing contracts) to pressure customers to move to NGSCB and the Microsoft applications associated with it.

Thus the policy enforcement mechanisms that Microsoft has employed will also suit their purposes to retain their market share in software. For example, Microsoft Word is currently the industry standard word processor. NGSCB would encourage the developers of Word to take advantage of its policy enforcement mechanisms in order to make sure that only Word applications could access Word documents, therefore locking consumers into using Word for these documents and not, as they can today, using rival software such as OpenOffice. In this case, Microsoft could easily hide behind the banner of neutrality, as they could claim that users chose to use Word, and so such policy enforcement is effectively under their (the users') control (i.e. if they don't want such enforcement, they don't have to use Word).

Microsoft is not the only company to attempt to distance itself from being seen to take sides in policy issues; that IBM and other Trusted Computing Group companies wish to avoid being associated with the potential DRM capabilities of Trusted Computing [Safford, 2002b] suggests that taking a firm stance on these issues is seen as detrimental to business. Consumers should be wary lest the evasion of responsibility at this stage means that the companies involved will also be able to avoid responsibility if things go wrong. For scientists in any field to distance themselves from accountability for the results of research by claiming neutrality is to walk a fine ethical line, and for scientists involved in large-scale consumer operations this line is even finer.

3.3 Digital Rights Management

One of the most discussed issues is the potential for Trusted Computing's encryption and attestation abilities to be used for Digital Rights Management (DRM). DRM, in this context, is the attempt to use computer technology to

enforce the property rights which accrue to holders of any form of copyright. For obvious economic reasons it is currently particularly important to the holders of large-scale portfolios of copyright in films and music. With previous attempts at DRM poorly implemented and mostly unsuccessful (see Chapter 2), Trusted Computing seems to offer a much greater chance of success in DRM, because of its potential not only to require specific systems to prove that they hold a licence before showing or copying copyrighted material (something which earlier copy-protection systems can also do) but also to stop *other* systems — systems which the copyright holder has not anticipated in any detail — from bypassing the protection mechanisms.

Although no company has claimed that DRM is a definite aim for their Trusted Computing ventures, many have suggested that the low-level encryption and attestation functions available on the TPM are well suited to DRM purposes [Anderson, 2002, Anderson, 2003, Schoen, 2003, Stallman, 2002]. Those who disagree with tarring Trusted Computing and DRM with the same brush argue one or more of the following points:

– Trusted Computing does not necessarily make DRM easier to employ.

– The combination of Trusted Computing and DRM is not necessarily going to be implemented.

– The criticisms of Trusted Computing in this context cast an unnecessary bad light on the potential good applications of DRM [Safford, 2002b, Anonymous, 2004].

Despite these claims, Digital Rights Management has long been a definite goal for Microsoft's NGSCB and very possibly still is. "It's a funny thing," remarked Microsoft Chairman Bill Gates, "We came at this thinking about music, but then we realized that e-mail and documents were far more interesting domains." [Thurrott, 2002] Digital Rights Management and NGSCB have always been intertwined, and the Trusted Computing Group's initiative, which is not entirely separate from NGSCB, will be at least somewhat affected by any publicity brought about by the resulting use of DRM with NGSCB. Good intentions or no, Trusted Computing Group members wishing to separate themselves from DRM issues will find it difficult when the majority of TCG TPMs are in NGSCB-controlled computers.

What has made public discussion of this problem difficult is not the separation of DRM and Trusted Computing, but placing the ideals of Trusted Computing in a realistic context that everyday users will likely find themselves if the technology is expanded and developed. Those who wish to alert people to the problems surrounding the technology must take examples that are plausible, not only technology-wise, but that are also within the realm that the companies involved have explored previously. We need not look further than Microsoft's Janus DRM software [Borland, 2004], software that adds time restrictions to media files, to realise that Digital Rights Management is a realm that technology companies are interested in, so any software or hardware that helps to restrict data usage is likely to be used for that purpose.

3.4 Software Lock-In and Competition Restriction

Another major concern about Trusted Computing involves the potential use of the technology to lock customers in to using a particular piece of software: in other words, to force them to use it for a substantial period — years or decades — after it ceases to be the best solution to their problems. Lock-in of this magnitude is common in the Information Technology world: for example, many banks still use programs which are expensive to maintain (in the sense of adding minor essential improvements to keep up with changes in legislation and other policies) because they are written in languages such as COBOL which have been obsolete (and hence expensive to hire experts in) for thirty years [Shapiro and Varian, 1998].

How does Trusted Computing encourage software lock-in? Using the Trusted Computing cryptography and attestation mechanisms in software not only makes encrypting important data possible, but could also allow software to restrict access to competitive products that wish to interoperate with it, such as "unofficial" versions of instant messenger clients [Anderson, 2002, Schoen, 2003]. It would be difficult for Trusted Computers to employ attestation in the simple form given in the TCG specifications for this sort of purpose, so each piece of software would have to govern its own interoperability standings, by, for example, issuing certificates from a centralised server tied to the Attestation Identity Key given to it by the licensee's computer. A software manufacturer could use the sealed storage capabilities of Trusted Computing to prevent other applications from accessing the data kept within, thus locking out interoperative applications within the one platform [Schoen, 2003]. Whether or not this is an actual goal of the Trusted Computing initiatives, it is definitely going to be a side-effect, as access policies for sealed storage would need to protect against third party applications accessing secret data.

How do proponents of Trusted Computing deal with this problem? Not surprisingly, they don't. Microsoft's recent antitrust cases [US Department of Justice, 2004] have meant that it has maintained a strictly low profile when commenting on software lock-in issues, and especially on the specific case of NGSCB. IBM's David Safford [Safford, 2002a] has countered accusations of Trusted Computing's aims for DRM, but has not attempted to address the problems of software lock-in, or even acknowledged their existence. Such companies could benefit greatly from lock-in on a software (such as Microsoft) or hardware (such as IBM) level, and would best serve their own interests by publicly ignoring the problem.

Most serious thought on countering the software lock-in problem has therefore come from disinterested commentators on Trusted Computing, most of whom are more or les opposed to both software lock-in and Trusted Computing in general. One of these people, Seth Schoen, despite misgivings about Trusted Computing as it stands, has suggested a solution to the software lock-in problem, "Owner Override".

3.5 Making Trusted Computing More Palatable

Several suggestions have been made to make Trusted Computing more appealing to those who find fault with it. Seth Schoen's aforementioned Owner Override approach is the major one, presenting a solution to the problem of the lack of owner control over the computer. In allowing a physically present user to override the information about the software state of their computer that would be sent to a computer requesting an attestation, the user would be able to send the requesting computer information of his or her choosing, rather than what is automatically generated by the computer. This would be similar in functionality to current Web browsers' capabilities to present themselves to version-checking Web sites as a different browser, in order to bypass any checking mechanism restricting use of that Website to a particular browser. Although this degrades the overall "magic bullet" appearance of Trusted Computing, by retaining security issues that are present now, Schoen claims it retains some of the more important parts of the Trusted Computing architecture, namely, informing the user as to whether the computer's software environment state has changed without the user's knowledge. This allows the computer owner to interoperate with a remote entity regardless of the client software they use, as they can submit a modified attestation to the requesting computer, making it seem as though it is running the required software [Schoen, 2003]. This would cut down on lock-in issues, but could open up networks to rogue clients intent on causing damage (much as such networks are open today). Although Owner Override would improve some parts of Trusted Computing, it would be likely to have several drawbacks:

- Owner Override might degrade the overall security of the system by making it more complex, and thus more difficult to control; for example, the sending of false information to remote entities would not be much use unless the generation of the information was at least partly automated (because the information would be complicated enough to be extremely burdensome for a human to memorise and type), and such automation could lead to serious security holes;
- Owner Override might instil a false sense of security in users of networked applications; and
- Owner Override would render most DRM (good or bad) applications ineffective.

On the whole, Owner Override is a good starting point for discussion about modifications to Trusted Computing, but it should not be seen as an overall solution to the issues Trusted Computing raises. And this is doubly so since, for the reasons of control mentioned above (and others omitted for lack of space) neither Microsoft nor any other member of the Consortium is likely to ever build a system incorporating Schoen's idea.

Kursawe and Stüble [Kursawe and Stüble, 2003] suggest a similar technique, allowing the owners to not only "substitute" the PCR values, much like Owner

Override, but also access and migrate the Storage Root Key and overwrite and re-generate the Endorsement Key, amongst other things, in certain circumstances. This brings with it all the drawbacks of vanilla Owner Override: although the arguments for giving the owner access to the SRK when the user is moving data from one computer to another (for example, for back-up) are sound, access to the SRK in other circumstances could be problematic for DRM-based applications, both the good ones and the bad.

There is nothing one can add to the current Trusted Computing specifications that would be an adequate compromise for all parties. Giving the owner access to keys used for encryption purposes would defy Digital Rights Management applications. Allowing the owner to change the PCR values would lead to problems with attestation in trusting reported values, and the potential for Trusted Computing to leave us no better off than we are now. It seems to me that the Trusted Computing initiative, while an excellent start in investigating the realms of securing the computer and network, is not going to be the best answer for many of the parties concerned, particularly typical end-users. Perhaps, however, the DRM applications are not necessary to the overall improvement to security that Trusted Computing would offer, and in this light, giving the user the keys to the system, and thus the ability to decrypt information stored on the system, would be appropriate in order to keep an even standing in the realms of interoperation and user control.

Conclusion

In this thesis I have investigated some of the advantages and disadvantages of Trusted Computing. I have explored both social and technological issues, including those of trust, rights to privacy, ownership and control, vulnerabilities, uptake, benefits and disadvantages, and potential uses and abuses. I have discussed some of the problems of definition that plague the project, and that will continue to cause confusion in both implementation and use unless directly addressed.

I have established that Trusted Computing, in its current state, will not be an effective long-term solution to security woes, and could in fact hinder efforts to stamp out illegal and anti-social activities as well as restrict honest consumers' rights to carry out legitimate activities.

Trusted Computing, as a blanket security system, is too restrictive, and yet if these restrictions were to be lifted (through Owner Override, for example) it would be largely ineffective except for certain low-level tasks which could easily be dealt with separately (such as memory protection). However, Trusted Computing is scheduled to be deployed with upcoming versions of operating systems, even though many problems, such as backup and restoration, have not yet been addressed at all, and others have been addressed only superficially and unsatisfactorily. Trusted Computing is a premature grab for acknowledgement in the security field which could easily backfire. This is particularly likely if the issues which have not yet been addressed are abstracted to the software level, as the issues with Denial of Service attack issues have been. Microsoft has already encountered problems requiring large-scale reassessment with its plans for NGSCB, and will need to pay much more attention to its customers to divine exactly what their security needs are, and to come to a compromise between these and the limits of the Trusted Computing architecture.

Despite having looked closely at the dialogue on the changes required for NGSCB to be acceptable to business clients, it is hard to know whether or not such a compromise will result in NGSCB being taken to pieces to reappear as a series of smaller-scale endeavours, but the issues raised in this thesis show that such a move would not be surprising.

The social issues that Trusted Computing is likely to encounter will only make dealing with the technical issues worse, as public relations material, education, and advertisements will attempt to display Trusted Computing as the solution to practically all current security problems, to the point where users are

likely to become lax with other important parts of computer security not covered by Trusted Computing, such as choice and storage of personal passwords. Modifications likely to be made to Trusted Computing will not necessarily improve the system at all, and even as a compromise may end up creating more problems than they solve. Overall, Trusted Computing will not, without major modifications to specifications and implementations, be able to deliver a solution to the problems of computer and network security without severely encroaching on a user's privacy and freedom of choice.

Glossary

Adware Software, usually installed by mistake or stealth on a victim's computer, that displays unsolicited advertising.

AIK Attestation Identity Key: A key created within the TPM, signed by the Endorsement Key, that is used for responding to attestation requests and other general TPM functions with a relative level of anonymity.

API Application Programmer Interface: Application Programmer Interface: A set of functions provided by one piece of software for the use of another piece of software, at a level suitable for programmers to use to make an interface between a new piece of software and an existing piece of software (typically an operating system).

Application Piece of software suitable for direct use by a user: for example, a word processing program.

Attestation The process by which a program can digitally sign and acknowledge that a piece of data was created within a secure operating environment in which the software running is known and can be identified.

ASCII American Standard Code for Information Interchange: A de facto standard for encoding text characters on a computer. Characters that are encoded are all upper- and lower-case Latin alphabet characters, numbers, and punctuation symbols.

Bandwidth The data transmission rate, i.e. the maximum amount of data that can be transmitted along a channel.

BIOS Basic Input/Output System: A set of instructions stored in read-only memory on the computer, that provides the initial sequence of commands when the computer is switched on.

Boot The process of initialising the computer and operating system.

CRTM Core Root of Trust for Measuring Integrity Metrics: The initial set of instructions called alongside the BIOS to establish a secure boot process.

DRM Digital Rights Management: A set of mechanisms that attempts to prevent restricted media licensees from deviating from the the owner's intended use.

EK Endorsement Key: The public key pair that uniquely identifies the computer's TPM.

File handle A low-level structure within the operating system that identifies files and facilitates access to them programatically.

Firewall A hardware and/or software buffer between an internal network and the Internet. It often enforces restrictions on data allowed in or out based on access policies.

Hash A mathematical formula that renders an arbitrary-length piece of data to a unique fixed-length piece of data.

Malware Malicious software which is usually installed by accident or stealth. Examples include data miners, which attempt to capture information about the user (such as browsing habits, etc.), password retrievers, or other sorts of discreet monitoring software.

Nexus The software interface for Microsoft's NGSCB.

NGSCB Next-Generation Secure Computing Base: Microsoft's TPM-based Trusted Computing solution. Formerly known as "Palladium".

Open Standard A specification that allows for any interested party to create compliant works without paying royalty or suffering discrimination.

Operating System Software that handles the execution of applications, and performs functions such as load balancing, scheduling, storage allocation, interfacing between devices and applications, and error handling.

Palladium See *NGSCB*.

Patch An update to a piece of software that does not require the administrator to download and/or install the whole application from scratch.

PCR Platform Control Register: A 160-bit storage location for hashes of information about the security state of the computer.

PRIVEK The private part of the Endorsement Key.

Process table In a multi-tasking environment, a table containing all the information needing to be saved for when the CPU performs a context switch from one process to another.

PUBEK The public part of the Endorsement Key.

Reverse-engineering The practise of deconstructing or disassembling hardware or software in order to find out how it works. Often this is with the aim to build a similar project that can interoperate with the original.

RSA Rivest, Shamir, and Adelman's cryptographic algorithm. See Chapter 2.

Seal The function performed by the TPM to securely write data to an encrypted location with the computer in a specific security state. See *unseal*.

SHA-1 Simple Hashing Algorithm: See *hash*.

Social Engineering A process by which a malicious entity coerces a victim into revealing information that they would not otherwise divulge. See Chapter 1, footnote.

Spyware Malicious software specifically with the intent to spy on a user's activities, often to extract passwords, bank account details, etc.

SRK Storage Root Key: The key used as a basis for the secure storage functionality, to encrypt and decrypt under the seal and unseal directives.

TCG Trusted Computing Group: A group of companies developing the Trusted Computing and TPM specifications. Formerly known as the TCPA.

TCP/IP Transmission Control Protocol/Internet Protocol: The communications protocols upon which the Internet is based.

TCPA Trusted Computing Platform Alliance: See *TCG*.

Telnet A protocol for remote computing across TCP/IP, that allows the client computer to establish a remote terminal on a server in order to carry out remote tasks on that server.

Timing analysis A process by which security mechanisms are attacked by monitoring the timing between keystrokes sent over encrypted protocols in order to determine some information about them (such as the length of the password, etc.).

TPM Trusted Platform Module: The chip soldered onto the motherboard that provides trusted computing functionality.

Unseal The process by which a TPM authenticates and decrypts securely stored information depending on the security state of the computer. See *seal*.

Unicode A 16-bit character set which aims to replace the 128 character ASCII standard in order to encompass the wide variety of characters available in different languages. There are 65536 different Unicode characters available.

User A computer user, that is, any person who sits down at a computer or a terminal.

van Eck phreaking A method of eavesdropping on a computer user from a distance by way of using specific hardware to pick up and interpret electromagnetic fields output by computer hardware, particularly monitors. Signals received are pieced back together by the hardware so that the image is retrieved and can be displayed to the attackers.

Virus A piece of malware that is inadvertently executed by a user (usually through social engineering manipulation of the user) that replicates itself in order to infect others and carries out some form of vandalism on the host computer.

Vernam one-time pad The traditional one-time pad as used in World War I, which provides a block of completely random data as long as the message to be encrypted. These "pads" must be used only once (as each successive use makes it easier for the encryption to be broken), and they must be exchanged via a secure channel (so that the originator and target both have copies). These days, public key cryptography (see Chapter 1) is used to transfer the pads.

WinHEC Windows Hardware Engineering Conference: A conference held annually by Microsoft with workshops and talks on Windows hardware engineering.

Worm A self-replicating piece of malware, similar to a virus, that is usually focussed solely on propagation rather than vandalism, although the latter is often a side-product of the former as network bandwidth is choked up with the worm's self-propagation attempts.

Bibliography

[Anderson, 2002] Anderson, R. (2002). Cryptography and Competition Policy – Issues with 'Trusted Computing'. Technical report, Cambridge University.

[Anderson, 2003] Anderson, R. (2003). 'Trusted Computing' Frequently Asked Questions: – TC/TCG/LaGrande/NGSCB/Longhorn/Palladium/TCPA. http://www.cl.cam.ac.uk/~rja14/tcpa-faq.html.

[Anonymous, 2004] Anonymous (2004). Interesting Uses of Trusted Computing. http://invisiblog.com/1c801df4aee49232/article/0df117d5d9b32aea8bc23194ecc270ec.

[Bechtold, 2003] Bechtold, S. (2003). The Present and Future of Digital Rights Management – Musings on Emerging Legal Problems. *Digital Rights Management*, LNCS 2770:597–654.

[Bellovin, 1989] Bellovin, S. M. (1989). Security Problems in the TCP/IP Protocol Suite. *Computer Communications Review*, (2):32–48. http://www.research.att.com/s̃mb/papers/ipext.pdf.

[Borland, 2004] Borland, J. (2004). Microsoft's iPod killer? *C—Net News*. http://news.com.com/2100-1027-5183692.html.

[Boutin, 2002] Boutin, P. (2002). Palladium: Safe or Security Flaw? *Wired News*. http://www.wired.com/news/antitrust/0,1551,53805,00.htm.

[Coursey, 2002] Coursey, D. (2002). Why we can't trust Microsoft's 'trustworthy' OS. *ZDNet*. http://www.zdnet.com.au/newstech/os/story/0,2000048630,20266389,00.htm.

[Delio, 2003] Delio, M. (2003). Are You a Good or a Bad Worm? *Wired News*. http://www.wired.com/news/infostructure/0,1377,60081,00.html.

[Diffie and Hellman, 1976] Diffie, W. and Hellman, M. E. (1976). New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654. http://citeseer.ist.psu.edu/diffie76new.html.

[Ferguson and Schneier, 2003] Ferguson, N. and Schneier, B. (2003). *Practical Cryptography*. Wiley Publishing, Inc., Indianopolis, Indiana.

[Fisher, 2004] Fisher, K. C. (2004). Apple's FairPlay DRM cracked. *Ars Technica*. http://arstechnica.com/news/posts/1081206124.html.

[Foucault, 1975] Foucault, M. (1975). *Discipline and Punish*. Random House.

[Grossman and Webb, 1991] Grossman, J. and Webb, K. (1991). Local food and nutrition policy. *Australian Journal of Public Health*, 4:271–277.

[Halderman, 2003] Halderman, J. A. (2003). Analysis of the MediaMax CD3 Copy-Prevention System. Technical Report TR-679-03, Princeton University. http://www.cs.princeton.edu/ jhalderm/cd3/.

[Hewlett-Packard, 2003] Hewlett-Packard (2003). *Trusted Computing Platforms: TCPA Technology in Context*. Prentice Hall PTR.

[Information Sciences Institute, 1981] Information Sciences Institute (1981). Internet Protocol, DARPA Internet Program, Protocol Specification. Technical Report RFC 791, University of Southern California, 4676 Admiralty Way, Marina del Rey, California 90291. http://www.faqs.org/rfcs/rfc791.html.

[Krill, 2003] Krill, P. (2003). Microsoft readies kit for security initiative. *InfoWorld*. http://www.infoworld.com/article/03/06/19/HNngscbtech_1.html.

[Kursawe and Stüble, 2003] Kursawe, K. and Stüble, C. (2003). Improving End-user Security and Trustworthiness of TCG-Platforms. Technical report, Saarland University. http://www-krypt.cs.uni-sb.de/download/papers/KurStu2003.pdf.

[Laboratories, 2003] Laboratories, R. (2003). Cryptographic Challenges: The New RSA Factoring Challenge: RSA-576 is factored! http://www.rsasecurity.com/rsalabs/node.asp?id=2096.

[Latour and Woolgar, 1986] Latour, B. and Woolgar, S. (1986). *Laboratory Life*. Princeton Univ. Press.

[Lemos, 2003] Lemos, R. (2003). What's in a name? Not Palladium. *C—Net News*. http://news.com.com/2100-1001-982127.html.

[Lord, 2004] Lord, T. (2004). Microsoft Drops Next-Generation Security Project. http://slashdot.org/articles/04/05/05/1520224.shtml.

[Manjoo, 2002] Manjoo, F. (2002). Can we trust Microsoft Palladium? *Salon.com*. http://www.salon.com/tech/feature/2002/07/11/palladium/index.html.

[McCullagh, 2000] McCullagh, D. (2000). Teen Hacking Idol Hits Big Apple. *Wired News*. http://www.wired.com/news/culture/0,1284,37650,00.html.

[Microsoft, 2003a] Microsoft (2003a). Microsoft NGSCB Technical FAQ. http://www.microsoft.com/technet/security/news/ngscb.mspx.

[Microsoft, 2003b] Microsoft (2003b). Next Generation Secure Computing Base. http://www.microsoft.com/resources/ngscb/default.mspx.

[Microsoft, 2003c] Microsoft (2003c). Next-Generation Secure Computing Base Product Information. http://www.microsoft.com/resources/ngscb/productinfo.mspx.

[Naraine, 2004] Naraine, R. (2004). Microsoft: Full Steam Ahead for Palladium. *InternetNews.com*. http://www.internetnews.com/ent-news/article.php/3350021.

[Patrizio, 2002] Patrizio, A. (2002). Codebusters Crack Encryption Key. http://www.wired.com/news/technology/0,1282,55584,00.html.

[Raymond, 1996] Raymond, E. S. (1996). *The New Hacker's Dictionary*. MIT Press, 3 edition.

[Rivest et al., 1977] Rivest, R. L., Shamir, A., and Adelman, L. M. (1977). A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Technical Report MIT/LCS/TM-82.

[Rooney, 2004] Rooney, P. (2004). Microsoft Shelves NGSCB Project as NX Moves to Center Stage. *CRN*. http://www.crn.com/sections/BreakingNews/dailyarchives.asp?ArticleID=49936.

[Safford, 2002a] Safford, D. (2002a). Clarifying Misinformation on TCPA. Technical report, IBM Research. http://www.research.ibm.com/gsal/tcpa/tcpa_rebuttal.pdf.

[Safford, 2002b] Safford, D. (2002b). The Need for TCPA. Technical report, IBM Research.

[Schoen, 2003] Schoen, S. (2003). Trusted Computing: Promise and Risk. Technical report, Electronic Frontiers Foundation. http://www.eff.org/Infra/trusted_computing/20031001_tc.php.

[Schuba, 2000] Schuba, C. L. (2000). Analysis of a Denial of Service Attack on TCP. Technical report, Purdue University. http://ftp.cerias.purdue.edu/pub/papers/diego-zamboni/schuba-krsul-kuhn-spaf-sundaram-zamboni-synkill.pdf.

[Sewell, 2002] Sewell, G. (2002). Peering Behind the Mask of Trust: Between Trust and Control in Contemporary Organisations. Technical report, University of Melbourne. http://www.management.unimelb.edu.au/research/wph5.pdf.

[Shapiro and Varian, 1998] Shapiro, C. and Varian, H. R. (1998). *Information Rules*. Harvard Business School Press.

[Stallman, 2002] Stallman, R. M. (2002). *Free Software, Free Society: The Selected Essays of Richard M. Stallman*. GNU Press.

[Stevens, 1994] Stevens, W. R. (1994). *TCP/IP Illustrated: The Protocols*, volume 1 of *Addison-Wesley Professional Computing Series*. Addison Wesley.

[Thurrott, 2002] Thurrott, P. (2002). Microsoft's Secret Plan to Secure the PC. *Windows Network and .Net Magazine*. http://www.winnetmag.com/Article/ArticleID/25681/25681.html.

[Trusted Computing Group, 2003] Trusted Computing Group (2003). TCG TPM Specification v1.2: Design Principles. Technical report.

[Trusted Computing Group, 2004] Trusted Computing Group (2004). Current members. Technical report. https://www.trustedcomputinggroup.org/about/members/.

[US Department of Justice, 2004] US Department of Justice (2004). Antitrust Case Filings: United States vs. Microsoft. Technical report. http://www.usdoj.gov/atr/cases/ms_index.htm.

[Ware, 1970] Ware, W. (1970). Security Controls for Computer Systems (U): Report of Defense Science Board Task Force on Computer Security. Technical report, The Rand Corporation, Santa Monica, CA.

[Webb, 2001] Webb, M. (2001). Windows Tips: Disable Balloon Tips. *TechTV*. http://www.techtv.com/screensavers/windowstips/story/0,24330,3362976,00.html.

[Whitten and Tygar, 1998] Whitten, A. and Tygar, J. D. (1998). Usability of security: A case study. Technical Report CMU-CS-98-155, Carnegie Mellon University. http://reports-archive.adm.cs.cmu.edu/anon/1998/CMU-CS-98-155.pdf.