Trusted Computing: Promise and Risk

By Seth Schoen

Introduction

Computer security is undeniably important, and as new vulnerabilities are discovered and exploited, the perceived need for new security solutions grows. "Trusted computing" initiatives propose to solve some of today's security problems through hardware changes to the personal computer. Changing hardware design isn't inherently suspicious, but the leading trusted computing proposals have a high cost: they provide security to users while giving third parties the power to enforce policies on users' computers against the users' wishes -- they let others pressure you to hand some control over your PC to someone else. This is a "feature" ready-made for abuse by software authors who want to anticompetitively choke off rival software.

It needn't be this way: a straightforward change to the plans of trusted computing vendors could leave the security benefits intact while ensuring that a PC owner's will always trumps the wishes of those who've loaded software or data onto the PC.

Redesigning PC hardware for security

There is a widespread perception that personal computer security is in an unfortunate state and that something must be done to fix it. There are many promising approaches to improving security --redesigning operating systems, changing programming methodologies, or altering the PC's hardware itself. It is well known that a comprehensive defense against the security threats faced by PC users will involve several approaches, not just one. An insecure system can't magically become "secure" with the addition of a single piece of technology.

Changes to the design of PC hardware are one useful tool among many for improving security. While hardware changes aren't a prerequisite for increased security, they're undeniably helpful -- for example, by providing a way to store private keys (and therefore the private documents protected by those keys) safely. One family of projects to add security to PCs through hardware changes is known as "trusted computing". This broad term includes a mix of initiatives by individual processor manufacturers and OEMs, along with two particularly well-known larger projects.

The first of these is an operating system project by Microsoft -- originally called Palladium and now referred to as the Microsoft Next-Generation Secure Computing Base, or NGSCB. The NGSCB project specifies software changes that take advantage of the security benefits made available by a planned new PC hardware design. The other well-known project is a hardware specification project run by a consortium originally called the Trusted Computing Platform Alliance, or TCPA. TCPA issued several specification documents and then changed its name to the trusted computing group, or TCG.

Between them, these two projects have created a bewildering array of new terminology, including the obligatory thicket of new acronyms. In several cases, one of these projects has devised many different names for a single concept -- even as the other project has its own entirely different terminology. A reasonably complete glossary for these two projects could fill dozens of pages. In the interest of simplicity, we note that the requirements of NGSCB are converging with the features of the design specified by TCG. (Microsoft is a TCG member and has expressed an interest in using the TCG design in the role of the hardware components required by NGSCB.) Some OEMs have begun to integrate early TCG chips onto their computers' motherboards; in the future, more computer manufacturers may include future versions of trusted computing circuits in their PCs. The NGSCB software would be one application of several which could take advantage of the features of these chips.

While these projects are still distinct, it is reasonable to speak of a single "trusted computing architecture" toward which both projects are headed. (Only a portion of this architecture is described by the most recently published TCG specification, and, as TCG notes, additional software will be required to make use of many of these features.) Less well known trusted computing projects under development by processor vendors (and TCG members) Intel and AMD may fill in some of the gaps between what TCG has so far specified and what NGSCB would require. Intel's LaGrande Technology (LT) and AMD's Secure Execution Mode (SEM), for example, provide hardware support needed for all the major feature groups in NGSCB. The Intel and AMD projects are not discussed as separate entities here, but their features would build on TCG features to provide the hardware support demanded by NGSCB.

One important similarity between the NGSCB design and the existing TCG specification is that both contain a "remote attestation" feature, which we will criticize extensively below. Even though there are differences between Microsoft's and TCG's technical descriptions of remote attestation, both can, given proper operating system support, be used in functionally equivalent ways. Whether or not the NGSCB and TCG projects converge on a single hardware design, the general criticisms of attestation here will properly apply to either.

We are describing a work in progress, but it is important that we start now to understand the proposed changes to the PC and their likely effects on our computing activities. Broadly speaking, the trusted computing architecture is a misguided implementation of a valuable idea, and would offer both advantages and disadvantages to computer owners. In Microsoft's account of the trusted computing architecture, the anticipated changes are divided at a high level into four groups, all of which require new hardware to be added to today's PCs. These are

1. Memory curtaining
2. Secure input and output

3. Sealed storage
4. Remote attestation

Each feature has a different security rationale, although the features can be used in conjunction with one another.

1. Memory curtaining

Memory curtaining refers to a strong, hardware-enforced memory isolation feature to prevent programs from being able to read or write one another's memory. Today, an intruder or malicious code can often read or alter sensitive data in a PC's memory. In the trusted computing design, even the operating system should not have access to curtained memory, so an intruder who gains control of the very operating system would not be able to interfere with programs' secure memory.

Although memory isolation can be achieved in software, this requires some combination of rewriting operating systems, device drivers, and possibly even application software. Implementing this feature in hardware instead permits greater backwards compatibility with existing software and reduces the quantity of software which must be rewritten. (In general, many of the security benefits of trusted computing could be achieved in some form simply by rewriting software, but this appears impractical to some.)

2. Secure I/O

Secure input and output, or secure I/O, aims to address the threats posed by keyloggers and screen-grabbers, software used by snoops and intruders to spy on computer users' activities. A keylogger records what you type, and a screen-grabber records what's displayed on the screen. Secure I/O provides a secure hardware path from the keyboard to an application -- and from the application back to the screen. No other software running on the same PC will be able to determine what the user typed, or how the application responded. (At the same time, secure I/O will provide protection against some more esoteric attacks. It will allow programs to determine whether their input is provided by a physically present user, as distinct from another program impersonating a user. And it will defeat some cases of forgery where one program attempts to corrupt or mask another's output in order to deceive the user.)

3. Sealed storage

Sealed storage addresses a major PC security failing: the inability of a PC to securely store cryptographic keys. Customarily, the keys and passwords that protect private documents or accounts are stored locally on the computer's hard drive, alongside the documents themselves. This has been compared to leaving the combination to a safe in the same room with the safe itself. In practice, intruders who break into a computer can frequently copy decryption and signing keys from that computer's hard drive. Since the keys must be accessible to computer users in order to be usable for their intended purpose, security engineers have faced a quandary: how can keys be stored so that they are accessible only to legitimate users and not to, say, a virus, which might acquire the same privileges as a legitimate user?

Sealed storage is an ingenious invention that generates keys based in part on the identity of the software requesting to use them and in part on the identity of the computer on which that software is running. The result is that the keys themselves need not be stored on the hard drive but can be generated whenever they are needed -- provided that authorized software tries to generate them on an authorized machine. If a program other than the program that originally encrypted, or "sealed", private data should attempt to decrypt, or "unseal", that data, the attempt is guaranteed to fail. Similarly, if the data is copied in encrypted form to a different machine, attempts to decrypt it will continue to be unsuccessful. Thus, your e-mail could be readable to your e-mail client, but incomprehensible to a virus. Sealed storage represents a clever solution to a previously intractable key storage problem.

For example, suppose you keep a private diary on your PC today. You want to prevent the diary from being moved off your computer without your permission, much as you might lock a paper diary inside a desk drawer. While existing access control and encryption systems address this goal, they might be bypassed or subverted. If someone compromises your system, or it becomes infected with a worm or virus, local software could be altered, or private documents could be e-mailed or copied to other computers. (The SirCam e-mail worm did precisely this -- whenever it infected a computer, it sent files it found there as e-mail attachments to randomly chosen Internet users. A substantial amount of private and confidential information was inappropriately disclosed as a result.)

You can encrypt your diary using a password, but if your password is short, someone who can copy the encrypted diary will still be able to decrypt it (by trying each possibility in a brute force attack). What's more, if the encryption software you use, or the editor in which you compose the diary, is surreptitiously replaced with a modified version, it might leak the decrypted diary's text (or your password) to a third party.

Sealed storage can work together with memory curtaining and secure I/O to ensure that your diary can only be read on your computer, and only by the particular software with which you created it. Even if a virus or worm like SirCam leaks your encrypted diary, the recipient will not be able to decrypt it. If an intruder or a virus surreptitiously alters your encryption software, it will no longer be able to decrypt the diary, so the contents of your diary will remain protected.

4. Remote attestation

Remote attestation is the most significant and the most revolutionary of the four major feature groups described by Microsoft. Broadly, it aims to allow "unauthorized" changes to software to be detected. If an attacker has replaced one of your applications, or a part of your operating system with a maliciously altered version, you should be able to tell. Because the attestation is "remote", others with whom you interact should be able to tell, too. Thus, they can avoid sending sensitive data to a compromised system. If your computer should be broken into, other computers can refrain from sending private information to it, at least until it has been fixed.

While remote attestation is obviously useful, the current TCG approach to attestation is flawed. TCG attestation conspicuously fails to distinguish between applications that protect computer owners against attack and applications that protect a computer against its owner. In effect, the computer's owner is sometimes treated as just another attacker or adversary who must be prevented from breaking in and altering the computer's software.

Remote attestation works by generating, in hardware, a cryptographic certificate attesting to the identity of the software currently running on a PC. (There is no determination of whether the software is good or bad, or whether it is compromised or not compromised. "Identity" is represented by a cryptographic hash, which simply allows different programs to be distinguished from one another, or changes in their code to be discerned, without conveying any sort of value judgment.) This certificate may, at the PC user's request, be provided to any remote party, and in principle has the effect of proving to that party that the machine is using expected and unaltered software. If the software on the machine has been altered, the certificate generated will reflect this. We will see that this approach, although elegant, proves problematic.

How trusted computing affects the PC

Each of these four feature groups is likely to be useful to computer security, because each can be used by appropriate software to prevent or mitigate real attacks currently used against PCs. Thus, a PC with hardware support for these features can provide security guarantees that might be difficult to offer without hardware support. Of course, flaws in software will still permit other attacks, including the disclosure of private information. Trusted computing technology can't prevent computer security holes altogether. In general, it seeks to contain and limit the damage that can result from a particular flaw. For instance, it should not be possible for a coding flaw in one application (like a web browser) to be abused to copy or alter data from a different application (like a word processor). This sort of isolation and containment approach is an important area of computer security research and is used in many different approaches to computer security, including promising techniques outside of trusted computing.

The trusted computing features just described will add new capabilities to the PC. To be used, they must be supported by software; in the absence of trusted computing software drivers, the trusted computing PC is just an ordinary PC, which remains capable of running all existing PC software. To put this another way, the trusted computing architecture is designed to be backwards-compatible in supporting the ability to run existing operating systems and application software. Microsoft also anticipates that future versions of Microsoft Windows (which could include NGSCB software) would be backwards compatible, able to run essentially all of today's DOS and Windows applications. In addition, the new PCs could run new trusted-computing-aware applications that take advantage of the new hardware features.

Misconceptions about trusted computing

Misconceptions about this design abound. The most common misconception denies that the trusted computing PCs would really be backwards-compatible or able to run existing software. While it is certainly possible for manufacturers to build non-backwards-compatible PCs, or PCs incapable of running particular code, nothing in the TCG specifications insists on this. More importantly, the trusted computing architecture security model does not require that insecure, harmful, or undesirable software be prevented from running. The security model instead concentrates on software isolation -- preventing running programs from interfering with one another.

When programs are adequately protected against interference by other programs, there is no security requirement that any particular software should be prevented from running. Just as multi-user operating systems allow users to run the software of their choice while protecting other users from the effects of that software, NGSCB could allow users to run the software of their choice while protecting other software from its effects. Only a particularly crude security model would require prohibiting "bad" software from a computer entirely, and the NGSCB model is not so crude. In addition, that approach would require some means of determining which software is "bad", which would truly be a daunting task. (Some proprietary systems assume that all software not signed by a recognized authority is "bad", but users would properly reject this heavy-handed approach in the computer environment. They rightly insist on being able to write and use software without the prior approval of some authority.)

None of the hardware demanded by NGSCB appears to be specific to Microsoft Windows. The TCPA/TCG hardware design is clearly not specific to any particular operating system. IBM researchers have recently published software under the GNU GPL to make a TCPA TPM chip work with the Linux kernel. This software is usable today to improve the security of cryptographic key storage on Linuxbased systems running on hardware that supports TCPA.

Neither TCG nor NGSCB would itself inherently prevent users from using any particular operating system, program, or data file. And neither inherently requires or includes a mechanism to spy on users.

Where's the problem?

It is clear that trusted computing hardware provides security benefits, if software is prepared to take advantage of it. But trusted computing has been received skeptically and remains controversial. Some of the controversy is based on misconceptions, but much of it is appropriate, since trusted computing systems fundamentally alter trust relationships. Legitimate concerns about trusted computing are not limited to one area, such as consumer privacy or copyright issues.

We have at least two serious concerns about trusted computing. First, existing designs are fundamentally flawed because they expose the public to new risks of anti-competitive and anti-consumer behavior. Second, manufacturers of particular "trusted" computers and components may secretly implement them incorrectly. We will discuss the first of these problems in greater detail here.

Problem: Third-party uncertainty about your software environment is normally a feature, not a bug

Even if the hardware is implemented according to published specifications, it could still be used in ways that harm computer owners. (Lucky Green, Ross Anderson)

Even as trusted computing architectures provide security benefits, they may include features that can be abused, to the detriment of the customers who are asked to adopt the technology. Chief among these features is remote attestation, which Microsoft describes as "break[ing] new ground in distributed computing" because no comparable feature exists in current computers.

Security design necessarily includes specifying a threat model: what kinds of attacks and what kinds of attackers is a security measure meant to prevent against? A security measure that prevents one attack may be completely ineffective against a different sort of attack; conversely, a security measure required for some purpose might be useless at best for those who do not share that goal. Our most fundamental concern is that trusted computing systems are being deliberately designed to support threat models in which the owner of a "trusted" computer is considered a threat. These models are the exception rather than the rule in the history of computer and communications security, and they are not part of the rationales for trusted computing publicly offered by its proponents.

Attestation is appropriate for the purpose of preventing the software on a computer from being changed without the knowledge of the computer's owner (for instance, by a virus). Unfortunately, the attestation model in TCG's current design can equally effectively prevent the software on a computer from being changed deliberately by the computer owner with his or her full knowledge and consent. While the owner is always free to alter software, attestation adds a new risk: doing so may now eliminate the computer's ability to interoperate with other computers.

Because third parties currently have no reliable way to tell what software you are using, they have no reliable way to compel you to use the software of their choice. This aspect of the status quo is almost always a benefit for computer owners -- in terms of improved competition, software choice, software interoperability, and owners' ability to control their computers -- and therefore no attestation scheme that changes this situation is likely to be beneficial to consumers. Examples of the problems with changing this part of the status quo appear below.

Examples of abuses of remote attestation

Let's consider a few concrete examples of how TCG's attestation approach can harm interoperability or be used against computer owners.

1. On the Web
A web site could demand a software attestation from people wishing to read it. If they declined to provide an attestation, the site would refuse to deal with them at all; if the attestation showed that they were using "unapproved" software, the site would likewise decline to interact with them. Only those who could produce a digital certificate proving that their computers' software was satisfactory to the remote site would be permitted to use it. And this certificate could be produced -- in the current TCG scheme of things -- only if its contents were accurate.

Today, there is no really reliable way to achieve this effect. Therefore, attempts to coerce users into using particular software are currently ineffective; web sites are hard-pressed to control what operating systems and applications their users can use. Reverse engineering allows the creation of competitive new software that works well with existing software and services, and therefore computer owners have real choice. It is effectively impossible to punish them for choosing to use software other than that favored by those they deal with. If they want to use a different web browser or a different operating system, they know that they are unlikely to be locked out by the services most important to them.

For instance, some of today's on-line banking services claim to "require" Microsoft's browser, but users of other software are readily able to instruct their browsers to impersonate Internet Explorer. As far as the bank is concerned, its customers are accessing the site with the browser it demanded, but the users are not locked into technology decisions dictated by the shortsightedness of their financial institutions.

In a widely publicized case, MSN, the Microsoft Network, briefly refused to serve web pages to non-Microsoft browsers. In the interim, users of competitive products were able to fool MSN into thinking they were running Microsoft browsers. This would be impossible in an environment of routine NGSCB-style remote attestations. By allowing a web site to lock out disfavored software this way, these attestations would let anyone with market power leverage that power to control our software choices.

Security has nothing to do with many sites' motivations for preventing the use of disfavored software. Indeed, their reasons may be entirely arbitrary. In some cases, a site operator wants to force you to use a particular program in order to subject you to advertising. By verifying your use of an "approved" client, the site can satisfy itself that you have been forced to view a certain number of advertisements.

2. Software interoperability and lock-in

Software interoperability is also at risk. A developer of a web server program, file server program, e-mail server program, etc., could program it to demand attestations; the server could categorically refuse to deal with clients that had been produced by someone other than the server program's publisher. Or the publisher could insist on licensing fees from client developers, and make its server interoperate only with those who had paid the fee. (It is similarly possible to create proprietary encrypted file formats which can only be read by "approved" software, and for which the decryption keys must be obtained from a network server and are extremely difficult to recover by reverse engineering.)

The publisher in this case could greatly increase the switching costs for its users to adopt a rival's software. If a user has a large amount of important data stored inside a proprietary system, and the system communicates only with client software written by the proprietary system's publisher, it may be extremely difficult for the user to migrate his or her data into a new software system. When the new system tries to communicate with the old system in order to extract the data, the old system may refuse to respond.

The Samba file server is an important example of interoperable software created through reverse engineering. Samba developers studied the network protocol used by Microsoft Windows file servers and created an alternative implementation, which they then published as free/open source software. Samba can be deployed on a computer network in place of a Windows file server, and Windows client machines will communicate with it just as if it were a Windows server. (Similarly, Samba provides the means to allow non-Windows clients to access Windows file servers.) Without competitive software like Samba, users of Windows clients would be forced to use Windows servers, and vice versa. But if software could routinely identify the software at the other end of a network connection, a software developer could make programs demand attestations and then forbid any rival's software to connect or interoperate. If Microsoft chose to use NGSCB in this way, it could permanently lock Samba out of Windows file services, and prevent any useful competing implementations of the relevant protocols except by specific authorization.

Similarly, instant messaging (IM) services have frequently tried to lock out their competitors' clients and, in some cases, free/open source IM clients. Today, these services are typically unsuccessful in creating more than a temporary disruption for users. An attestation mechanism would be a powerful tool for limiting competition and interoperability in IM services. Some client applications could be permanently prevented from connecting at all, even though they offer features end-users prefer.

These are examples of a more general problem of "lock-in", often practiced as a deliberate business strategy in the software industry, to the detriment of business and home computer users alike. Unfortunately, the TCG design provides powerful new tools to enable lock-in. Attestation is responsible for this problem; sealed storage can exacerbate things by allowing the program that originally created a file to prevent any other program from reading it. Thus, both network protocols and file formats can be used to attack software interoperability.

3. DRM, tethering, forced upgrades, and forced downgrades

Many people have speculated that trusted computing technology is a way of bringing digital rights management (DRM) technology to the PC platform. Some portions of the trusted computing research agenda have roots in DRM, and Microsoft has announced a DRM technology (Microsoft Rights Management Services) that it says will make use of NGSCB. However, trusted computing developers deny that DRM is the main focus of their efforts, and trusted computing is useful for many applications besides DRM. Ultimately, DRM is just one of several uses of a technology like NGSCB -- but it illustrates the general problem that NGSCB's current approach to attestation tends to harm competition and computer owners' control.

The NGSCB design's elements can all be useful to implementers of DRM systems. Curtaining prevents information in decrypted form from being copied out of a DRM client's memory space, which prevents making an unrestricted clear copy. Secure output can prevent information displayed on the screen from being recorded, which prevents the use of "screen-scrapers" or device drivers that record information rather than displaying it. Sealed storage allows files to be stored encrypted on a hard drive in such a way that only the DRM client that created them will be able to make use of them. And remote attestation can prevent any program other than a publisher-approved DRM client from ever receiving a particular file in the first place.

Among these elements, remote attestation is the linchpin of DRM policy enforcement. If a remote system lacks reliable knowledge of your software environment, it can never have confidence that your software will enforce policies against you. (You might have replaced a restrictive DRM client with an ordinary client that does not restrict how you can use information.) Thus, even though other NGSCB features aid DRM implementations, only remote attestation enables DRM policies to be instituted in the first place, by preventing the substitution of less-restrictive software at the time the file is first acquired.

Other consumer-unfriendly software behaviors which can be implemented by means of attestation, combined with sealed storage, include tethering (preventing a program or a file from being migrated from one computer to another), forcing software upgrades or downgrades, and enabling some limited classes of "spyware" -- in this case, applications that phone home to describe how they are being used. (Some of these behaviors might be good things if they occur at a computer owner's behest, but not if they occur at a software publisher's or service provider's whim. For example, you might want to prevent a sensitive file from being moved off your computer, but you wouldn't want other people to be able to prevent you from moving your own files around.) Although all these unfriendly behaviors can be implemented in software today, they can in principle be defeated by well-understood techniques such as running a program in an emulated environment, or altering it to remove the undesirable behavior. Remote attestation makes it possible for the first time for a program to obtain and communicate reliable evidence about whether it is running in an emulator or whether it has been altered.

More generally, attestation in the service of remote policy enforcement leads to a variety of mechanisms of "remote control" of software running on your computer. We emphasize that these remote control features are not a part of NGSCB, but NGSCB does enable their robust implementation by software programmers. Lucky Green provides the example of a program written to receive from some authority a "revocation list" of banned documents it is no longer permitted to display. This mechanism would have to have been implemented in the software when it was initially written (or it would have to be added through a forced upgrade). If such a restriction were implemented, however, it would be essentially impossible for the user to override. In that case, some authority could remotely revoke documents already resident on computers around the world; those computers would, despite the wishes of their owners, comply with the revocation policy. The enforcement of this policy, like others, against the computer owner is dependent on the remote attestation feature.

4. Computer owner as adversary?

The current version of remote attestation facilitates the enforcement of policies against the wishes of computer owners. If the software you use is written with that goal in mind, the trusted computing architecture will not only protect data against intruders and viruses, but also against you. In effect, you, the computer owner, are treated as an adversary.

This problem arises because of the attestation design's single-minded focus on accurately reflecting the computer's state in every situation -- making no exceptions. A computer owner can disable attestation entirely, but not cause an attestation that does not reflect the current state of her PC -- you can't fool your bank about what browser you're using or to your other PC about what kind of Windows file sharing client you're running. This approach benefits the computer owner *only when the remote party to whom the attestation is given has the same interests as the owner*. If you give an attestation to a service provider who wants to help you detect unauthorized modifications to your computer, attestation benefits you. If you're required to give an attestation to someone who aims to forbid you from using the software of your choice, attestation harms you.

A user-centered, pro-competitive approach to attestation features would give the owner the power to guarantee that attestation is never abused for a purpose of which the owner disapproves, maximizing computer owners' practical control over their computers in real-world network environments.

Some trusted computing developers insist that their existing approach to attestation is reasonable because giving an attestation is voluntary. In every situation, they argue, you can decline to give an attestation if you prefer not to present one. (Indeed, TCG's design allows you to turn the TCG TPM chip off entirely, or decide whether to present an attestation in a particular situation.) But as we've seen, attestation can be used to create barriers to interoperability and access, so users will face an enormous amount of pressure to present an attestation. It's economically unreasonable to assume that a technology will benefit people solely because they can decide whether to use it.

We are not saying that the ability to communicate information about a computer's software environment is undesirable. This capability might well be useful for some security applications. We simply observe that the content of information about a computer's software environment should always be subject to the close control of that computer's owner. A computer owner -- not a third party -- should be able to decide, in her sole discretion, whether the information acquired by a third party will be accurate. This ensures that the attestation capability will not be used in a way contrary to the computer owner's interest.

A solution: Owner Override

The lack of computer-owner control of the content of attestations is the central problem with the current trusted computing proposals. It is an unacceptably grave design flaw that must be remedied before the trusted computing architecture as a whole package will be of clear benefit to computer owners.

A simple measure we call Owner Override could fix the problem by restoring others' inability to know for certain what software you're running -- unless you decide you would be better off if they knew. Owner Override subtly changes the nature of the security benefit provided by attestation. Currently, attestation tells remote parties whether the software on your computer has been changed. Attestation plus Owner Override would let remote parties know if the software on your computer has been changed *without your knowledge*. Thus, detection of illicit activity would still be practical. If, however, you had made deliberate changes on your own computer, you could conceal them, just as you can today, to prevent someone else from using your choices as a reason to discriminate against you.

Owner Override works by empowering a computer owner, when physically present at the computer in question, deliberately to choose to generate an attestation which does not reflect the actual state of the software environment -- to present the picture of her choice of her computer's operating system, application software or drivers. Since such an attestation can only be generated by the computer owner's conscious choice, the value of attestation for detecting unauthorized changes is preserved. But the PC owner has regained fine-grained control, even in a network environment, and the PC can no longer be expected to enforce policies against its owner. Owner Override removes the toolbox that allows the trusted computing architecture to be abused for anti-interoperability and anti-competitive purposes. It restores the important ability to reverse engineer computer programs to promote interoperability between them. Broadly, it fixes trusted computing so that it protects the computer owner and authorized users against attacks, without limiting the computer owner's authority to decide precisely which policies should be enforced. It does so without undermining any benefit claimed for the TCG architecture or showcased in Microsoft's public NGSCB demonstration. And it is consistent with TCG's and most vendors' statements about the goals of trusted computing.

(In a corporate setting, the corporation might be the owner of the computers its employees use, retaining the power to set network computing polices for its users. Since Owner Override requires users to provide owner credentials before defeating policies, it does not impair a computer owner's control over authorized users. A corporation can, for example, still use attestations to control what software employees can use on corporate desktop machines when they access corporate network resources.)

Owner Override does preclude some interesting new applications, particularly in distributed computing. In the status quo, it's not typically possible to send data to an adversary's computer while controlling what the adversary can do with it. Owner Override preserves that aspect of the status quo, to the regret of application developers who would like to be able to trust remote computers even while distrusting their owners. Similarly, Owner Override prevents trusted computing from being used to stop cheating in network games. Since Owner Override -- like trusted computing in general -- removes no existing features or functionality from the PC, we believe that its advantages significantly outweigh its disadvantages.

Despite the plausible desirability of hardware improvements to enhance computer security, not just any set of hardware changes will do. PC owners should think carefully about which direction they want their platform to develop. Trusted computing systems that protect your computer against you and prevent you from overriding policies are, on balance, a step backward. An Owner Override feature or its equivalent is a necessary fix to the design of trusted computing systems.

(This table shows how Owner Override preserves most security benefits of remote attestation while avoiding its risks.)

	Status Quo	**Attestation**	**Attestion + Owner Override**
Pros	Competition and interoperability are the norm User control and choice are protected Lock-in and remote control are difficult because computer owners have substantial control over all local software in all circumstances	Compromise of software (e.g., by a virus) can be made detectable by a remote party, which can act on this information Cheating in network games can be prevented, and distributed applications (Distributed.net, SETI@Home, etc.) can run on computers owned by untrustworthy parties without risking integrity of calculations or confidentiality of data Organizations can more	Compromise of software can still be made detectable by a remote party An organization can more effectively enforce policies against its own members, so long as they are using computers owned by the organization Computer owners retain substantial control over local software Competition, interoperability, user control and choice are

		effectively enforce policies against their own members "Paternalist" security policies that protect users from the consequences of certain of their own mistakes can be implemented	preserved
Cons	There is no way in general to allow a remote party to detect whether, without the computer owner's knowledge, local software has been inappropriately modified Cheating in network games cannot be prevented Cheating by unscrupulous participants in distributed computing projects cannot be prevented "Paternalist" security policies that protect users against their own mistakes are difficult to enforce	Third parties can enforce policies against computer owner where traditionally these would not have been technologically enforceable, or would have been enforceable only with difficulty -- for example: • DRM • application lock-in • migration and back-up restrictions • product activation • product tethering • forced upgrade • forced downgrade • application-specific spyware • preventing reverse engineering, etc.	Cheating in network games or by unscrupulous distributed computing participants still cannot be prevented "Paternalist" security policies remain difficult to enforce To the extent that computer owners might potentially benefit from the robust enforcement of DRM policies, they would not obtain those benefits

Problem: Verification of implementations

How can computer owners know that their trusted computing hardware has been implemented according to its published specifications? (Ruediger Weiss)
This is an important problem for *all* cryptographic hardware, not just trusted computing hardware. But since most PCs have not previously contained any specialized cryptographic hardware, most PC users simply haven't had occasion to worry about this

problem in the past. While any hardware could contain back doors or undocumented features, cryptographic hardware is unique in that it has access to important secret information as well as opportunities to leak that information through undetectable covert channels (for example, in attestation certificates). Thus, it is important to assure that trusted computer hardware manufacturers implement the specifications correctly, without including undocumented features that would allow them or third parties to obtain unauthorized access to private information.

Conclusion

We recognize that hardware enhancements might be one way to improve computer security. But treating computer owners as adversaries is not progress in computer security. The interoperability, competition, owner control, and similar problems inherent in the TCG and NCSCB approach are serious enough that we recommend against adoption of these trusted computing technologies until these problems have been addressed. Fortunately, we believe these problems are not insurmountable, and we look forward to working with the industry to resolve them.