

# Understanding Trusted Computing

## Will Its Benefits Outweigh Its Drawbacks?

Few pieces of vaporware have evoked a higher level of fear and uncertainty than Microsoft's Palladium: Palladium is a plot to take over cyberspace; Palladium will keep us from running any software not personally approved by Bill Gates; and so on. Its press got so bad that Microsoft

### Authenticated boot

An authenticated boot service monitors what operating system software was booted on the computer and gives applications a sure way to tell which operating system (OS) is running. It does this by adding hardware that keeps a kind of audit log of the boot process.

An OS normally boots in stages. First, a small piece of code in the Boot ROM executes. The Boot ROM code reads in more code from the Boot Block of the hard drive and executes that new code. In turn, the Boot Block code reads in a bigger chunk of code and executes that. This process continues, with each piece of code bringing in a larger piece, until the entire OS is up and running.

TC hardware keeps a tamper-evident log of this boot process, using a cryptographic hash function to detect any tampering with the log. For all practical purposes, the cryptographic hash is sufficient to uniquely identify the code, so any tampering with the log will change the hash, allowing the tampering to be detected.

When the machine starts booting, the TC hardware computes the cryptographic hash of the code in the Boot ROM and it writes that hash into the tamper-resistant log. Before it brings in the next block of code, the code from the Boot ROM computes the hash of the next block and appends it to the end of the tamper-resistant log. In turn, each chunk of code adds to the log the hash of the next chunk that will load. This process continues until the entire OS

EDWARD W. FELTEN  
Princeton University

changed the name to the mnemonically challenged "Next-Generation Secure Computing Base."

A similar, multicompany (Compaq, HP, IBM, Intel, and Microsoft), effort known as the Trusted Computing Platform Architecture (TCPA) garnered less abuse, but many people are suspicious of it as well. In fact, TCPA and Palladium have similar (though not identical) architectures and similar goals. Both systems are part of a more general approach called trusted computing (TC).

In this article I'll introduce TC's basic concepts and discuss their implications. However, the individual proposals are still in flux and some kind of convergence between them seems likely, so I'll not zero in on the details of any one proposal. Instead, I will discuss the general features of TC.

Rather than look at worst-case scenarios, I will assume that TC systems will be designed in ways their advocates predict. In other words, TC systems will have no coercive features beyond those absolutely necessary to make TC possible at all. This seems to me the most likely scenario, given the promises made by the vendors and

the public pressure on them not to renege on those promises.

TC systems do much less than their most strident opponents claim. They provide a simple, elegant, and powerful mechanism that has many uses. However, that mechanism might ultimately have a harmful impact on the software market—though not in the way that most of its opponents claim—because the main danger does not come from TC itself, but from what applications might do with that.

### What is TC?

TC operates through a combination of hardware and software: manufacturers add a little bit of new hardware to each computer to support TC functions, and then a special TC operating system mediates between the hardware and any TC-enabled applications. (It's worth noting that Microsoft envisions its TC operating system alongside Windows on the same computer—which provides some backward compatibility advantages—but does not impact the issues discussed in this article.)

TC provides two basic services, *authenticated boot* and *encryption*, which are designed to work together.

is booted, at which point the tamper-resistant log contains a record that can establish exactly which version of which OS is running.

### ***Certifying a configuration***

It is helpful for the TC hardware to know via its tamper-evident log what software configuration is running on a machine. But things get a lot more interesting when the TC hardware makes the configuration known to others. The TC hardware does this by producing a digital certificate that vouches for the configuration. It uses a private cryptographic key that only that individual machine's TC hardware knows; the TC hardware uses this secret key to sign a certificate that contains the configuration information, together with a random challenge value provided by the requester.

A configuration certificate can be presented to any recipient—to the user, for instance, or to a program running on another computer—and the recipient (provided that it generated the random challenge) can verify that the certificate is valid and up-to-date, so it can know what the machine's configuration is.

### ***Certifying applications***

In practice, certifying the OS's configuration is of some use, but there is an even stronger desire to certify the presence and configuration of application programs. TC allows this too.

Application configurations are certified by a two-layer process: First, TC can certify that a known OS version is running, and then that OS can certify the application's precise configuration. If you trust TC and the OS (knowing what version it is), then you can be confident that you know the application's configuration.

Configuration certificates have many uses, some of which may be immediately obvious, but a full discussion of their implications will have to wait until we have covered TC's other main feature.

### ***Encryption service***

The second major part of TC is an encryption service. This lets data be encrypted in such a way that it can be decrypted only by a certain machine, and only if that machine is in a certain configuration.

This service is implemented by a combination of hardware and software facilities. The hardware maintains a “master secret key” for each machine, and it uses the master secret to generate a unique secret encryption key for every possible configuration of that machine. As a result, data encrypted for a particular configuration cannot be decrypted when the machine is in a different configuration.

As with the certification of configurations, this service can be extended up to the application level by the OS. The resulting service lets data be encrypted so that it can be decrypted only by the desired version of the desired application, when running on top of the desired version of the desired OS, on the desired machine. We can use this facility to transmit data to a remote machine in such a way that the data can be decrypted only if the remote machine is in a certain configuration. Alternatively, an application can encrypt its own data before writing that data to disk, so that the data can

it is important to understand what TC will do, let's recall what TC will not do:

- Nothing in TC needs to restrict which OS your computer can run.
- Nothing needs to restrict which application software you can use.
- There is no need for a centralized bureaucracy to approve or disapprove OSs or applications.

TC as I have described it can work perfectly well with any OS and any application, whether open-source or proprietary. In fact, the direct effects of TC are relatively harmless. It is the indirect effects that are worrisome—not what TC does, but what others will build on top of it.

### ***TC and interoperation***

The principal effect of TC will be to give application software control over which other software can interoperate with it. For example, suppose you use your favorite word-processor program to create a text document. When you save that document to disk, the word processor can encrypt the document before writing it to disk, using the TC encryption facility to encrypt it in such a way that only the same application program can decrypt it.

**The principal effect of TC will be to give application software control over which other software can interoperate with it.**

be decrypted only by the same version of the same application, running on the same machine.

### ***How TC will change the rules***

If TC becomes widespread, it will change the ground rules of computing in significant ways. So, while

This use of encryption might keep the file's text out of the hands of an adversary who can run other programs on your computer. TC will keep these other programs from successfully decrypting your file—and that can be a good thing if the other programs are viruses or other types of malicious code. This is the major

promoted benefit of TC.

But note that the very same TC encryption that protects you from viruses also keeps you from opening the file with a different word-processor program, even if you want to do so. Remember, the file is encrypted so that the program that created it is the only one that can decrypt it. The only way to get the file's contents into a different program is to ask the first program to decrypt the data and transfer it. If the first program can't or won't do this, you're out of luck.

What this means is that the first program gets veto power over the second program's ability to interoperate with it. Previously, the second program's author could interoperate

whenever desired, as long as the first program's file format was known. But TC changes the rules so that no program has to interoperate if it doesn't want to. Every program gets absolute control over which other programs can interoperate with it, and exactly what degree of interoperation to allow in each case. It can interoperate with nobody, everybody, or only its "friends."

The same scenario applies when the two programs are communicating over the Internet, rather than running on the same machine. If, say, your Web browser tries to retrieve a page from my Web server software, then my server can use TC to encrypt the returned page in such a way that only a certain browser program can decrypt it. In other words, my server has veto power over your client's ability to interoperate with it. The trick works the other way too: your client can refuse to interoperate with my server if it doesn't like the server software I am running.

As in the single-machine case, in today's world without TC, one vendor can reverse-engineer another vendor's product and figure out how to interoperate with it, whether or not the second vendor approves. But again, TC changes the rules so that this cannot happen without the second vendor's approval.

### *The economics of TC*

Of course, the fact that one application can refuse to interoperate with another does not mean that it will refuse. Whether it does refuse will depend on its producer's economic interests. If the producer can make more profit by interoperating, it will do so; if it can make more profit by refusing to interoperate, it will take that course instead.

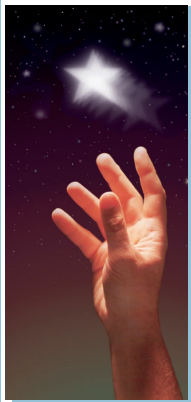
The economics of interoperation are subtle and sometimes counterintuitive. Customers generally like interoperability, so at least some vendors will choose to provide it. But there is at least one obvi-

ous case where a vendor would choose not to interoperate. If people use a product to communicate with other people, then everybody will want to buy a product that interoperates with the ones their colleagues and friends use—there will be a network effect. In a market with network effects, a dominant company might choose not to interoperate, as a way of locking consumers into their dominant product. For example, if one company had 95 percent of the market for email readers, then it would have an incentive to ensure that its email reader would not interoperate with others, because the lack of interoperability would give new buyers no real alternative but to buy the dominant product. Allowing interoperation would give new products a way to build market share gradually, but refusing to interoperate would doom them.

Without TC, the makers of new products probably would be able to interoperate, if they were willing to work hard enough to do so. TC makes that impossible, and so it changes the dynamics of the market in a way that threatens to reduce competition and retard the introduction of new products.

**W**hether the security benefits of TC outweigh its drawbacks is a difficult question. If we can reduce the level of vitriol in the TC debate and focus on the real effects of TC, perhaps we can answer that question before it is too late. □

*Edward W. Felten is an associate professor of computer science at Princeton University, and codirector of the Secure Internet Programming Laboratory at Princeton. His research specialties include computer security and privacy, and the interaction of law, policy, and technology. He received a PhD. in computer science and engineering from the University of Washington. Contact him at the Dept. of Computer Science, Princeton University, 35 Olden Street, Princeton NJ 08544; felten@cs.princeton.edu.*



REACH  
HIGHER

**Advancing in the IEEE Computer Society can elevate your standing in the profession.**

**Application to Senior-grade membership recognizes**

✓ 10 years or more of professional expertise

**Nomination to Fellow-grade membership recognizes**

✓ exemplary accomplishments in computer engineering

**GIVE YOUR CAREER A BOOST**

**UPGRADE YOUR MEMBERSHIP**

[computer.org/join/grades.htm](http://computer.org/join/grades.htm)