Trustworthy Computing

(TCPA, Palladium, etc....)
The Trusted Computing Group (TCG) is an alliance of Microsoft, Intel, IBM, HP and AMD which promotes a standard for a `more secure' PC. Other names you may see include TCPA (TCG's name before it incorporated), Palladium (the old Microsoft name for the version due to ship in 2004) and NGSCB (the new Microsoft name)

Original Motivation = Digital Rights Management (DRM)

DRM systems will try to hide the key in a complicated manner so that you still need whoever is providing the software(although if you find the key to decrypt the software you never need to use/talk to the software provider again).

Ex:
You have a Server and a Client /

Server wants to run part of the service on the client.

Problem is we don’t trust the client. (Platform owner)

If there was a secure chip in the client that could prevent and enforce certain rules.

**Digital Signatures**

Signer =
- message M
- secret key s
- public key p
- Signer will compute = \( C_{Sign}(m) \)
- Signer will send \( m, c \) (signature is unique/specific to the message)

Verifier =
- has public key p
- Verifier will run \( Verify_p(m, c) \) => will be valid or invalid (should be valid if signer is valid)
TCPA Chip generates:

- Secret Key S (only tcpa chip has access)
- Public Key P
- PCR (platform configuration register)

Extend Function

Extend( addr1, addr2)

\[
\text{PCR} \leftarrow H(\text{PCR}, \text{mem}(\text{addr1}, \ldots, \text{addr2}))
\]

Attest(void)

\[
\text{Return } \text{Sig}_S(\text{PCR})
\]

Ex

Suppose you want to convince someone you’re running windows 7. When you boot up windows 7/ windows 7 stuff will be put into ram / Eventually the windows 7 stuff will call and extend on all of ram / the PCR will hold a hash of what the RAM looks like at the time / now windows has a signature under the secret key of the memory contents on the tcpa chip.
This process is called Remote Attestation (simplified version).

The key of the TCPA chip in this instance is to prove to an external entity that you're running a certain piece of software.

Normally you'd establish a stack of software:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>WMP</strong> (Extend into OS kernel)</td>
<td></td>
</tr>
<tr>
<td><strong>OS Kernel</strong> (Perform Extend into PCR)</td>
<td></td>
</tr>
<tr>
<td><strong>Boot Loader</strong> (Perform Extend into PCR)</td>
<td></td>
</tr>
<tr>
<td><strong>Bios</strong> (Perform Extend into PCR)</td>
<td></td>
</tr>
</tbody>
</table>

There are several levels of trust here:

- Dell / the TCPM chip on the client/ the Software manufacturer.

**Issues:**
- Big trusted computing basis (TCB)
- Complex to support software version/variants/configs
- Big Brother
- Lock – IN
**Flicker (not the photo service)**: uses TCPA chip

**SKINIT** (addr, size)
- Disables DMA and other stuff (to prevent hardware attacks from below)
- Does a special extend on the PCR: \( \text{Extend}' (\text{addr}, \text{addr+size}) \)
- Jmp to addr

What the \( \text{Extend}'(\text{addr1}, \text{addr2}) \) does is:

\[ \text{PCR} \leftarrow \text{H} (\text{magic value}, \text{mem (addr1, .......,addr2)}) \]

[instead of putting the current PCR value it puts a magic value sort of the way the PCR starts with 0 as its default value]

Ex: Movie playing service sends you a chunk of code which takes a single encrypted video frame, decrypts it and displays it on the screen.

\[ \text{SKINIT} \]

\[ \text{Encrypted Frame} \]

\[ \text{Play 1 Frame} \]

\^\^ This solves vendor lock –IN and reduces our TCB.