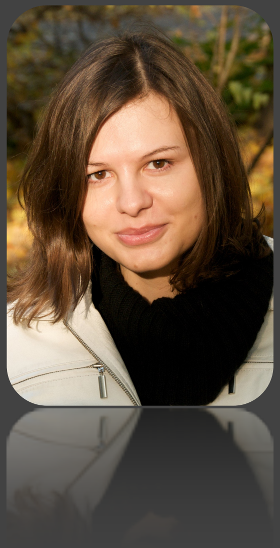
Thoughts about Trusted Computing

Joanna Rutkowska

INVISIBLE THINGS LAB

Confidence, Krakow, Poland, May 15-16, 2009
EuSecWest, London, UK, May 27-28, 2009

The Invisible Things Lab team:



Joanna Rutkowska

Alexander Tereshkin

Rafal Wojtczuk

Our recent research:

- ✓ **Vista Kernel Protection** bypass (2006, 2007)
- ✓ **BluePill** w/ **Nested virtualization** (2006-2008)
- ✓ **Xen hypervisor** compromises (2008)
- ✓ Chipset/CPU security bypass: **SMM attacks** (2008, 2009)
- ✓ **Intel TXT** bypass (2009)

1. TC’s basic **building blocks**
2. Practical **examples** of TC
3. Theory vs. **reality**

Trusted Computing

Goal: more secure desktop computers!

Solution:
A **hardware** element responsible for *checking* all (or part) of the software running on this platform

Requires software that is TC-aware!
(just the fact you buy TC-compatible hardware, doesn't make your system automatically more secure)

http://www.trustedcomputinggroup.org/

Members: AMD, Intel, IBM, Microsoft, Sun, Lenovo, HP, ...

Basic Building Blocks

Building Block #1: Trusted Platform Module (TPM)

The core component of TC

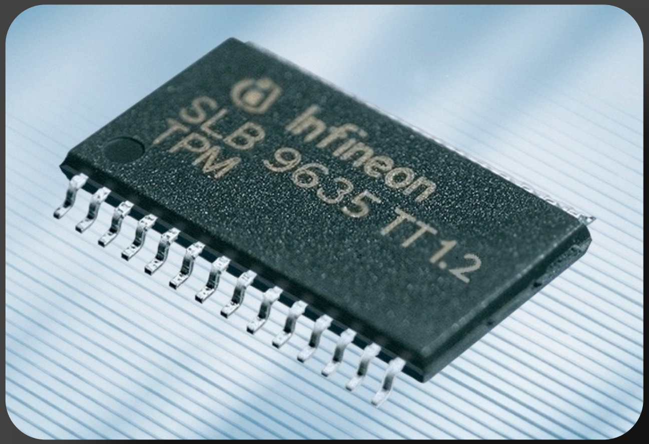
TPM 1.2

✓ **Passive** I/O device (master-slave)

✓ Special Registers: PCR[0...23]

✓ Interesting Operations:

- Seal/Unseal,
- Quote (Remote Attestation)
- some crypto services, e.g. PRNG, RSA, key gen



PCR registers

TPM 1.2 has at least 24 registers that can hold 160-bit values

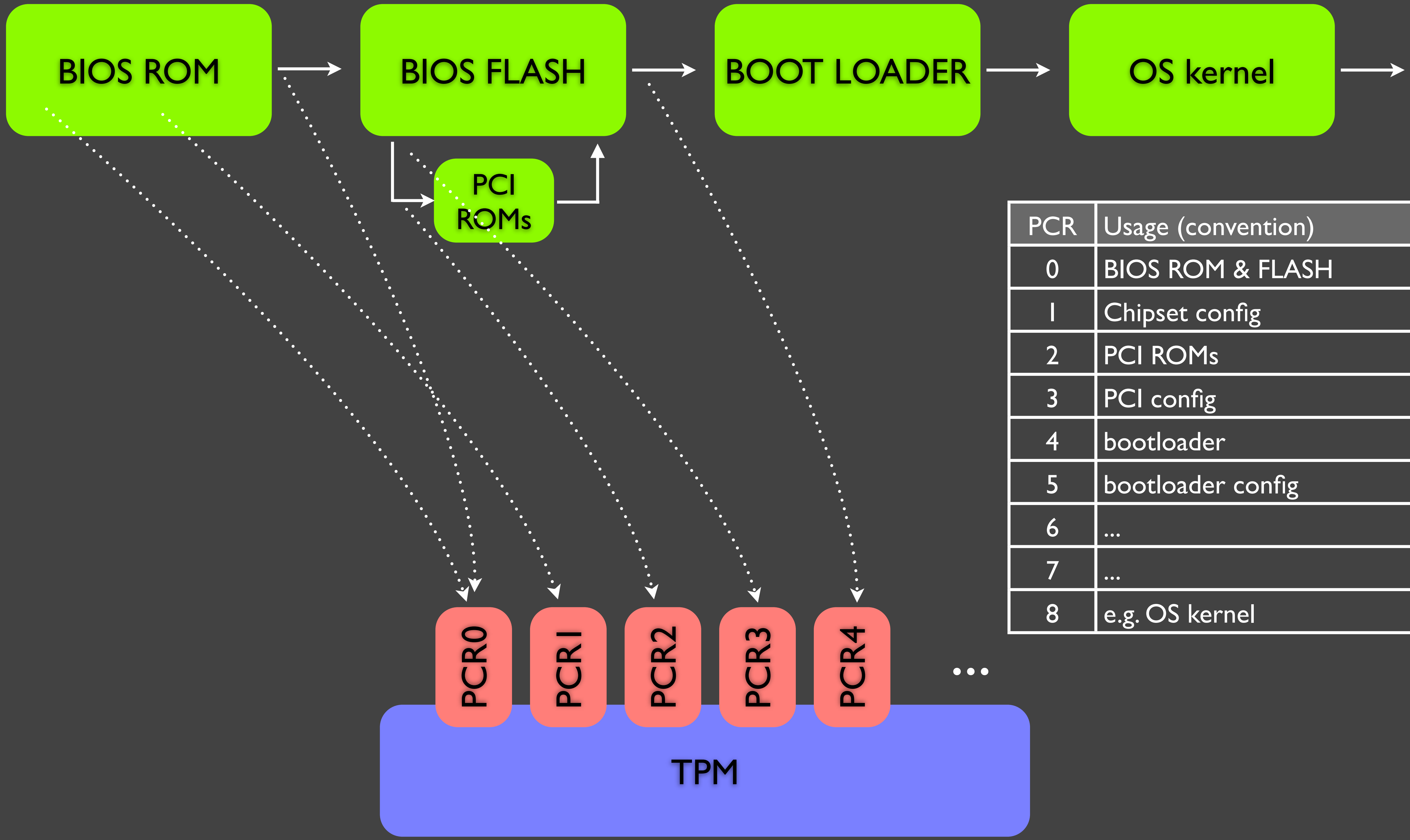
PCR “extend” operation

$$PCR_{N+1} = \text{SHA-1}\ (PCR_N \mid Value)$$

- A single PCR can be extended multiple times
- It is *computationally infeasible* to set PCR to a specified value
- $(ext(A), ext(B)) \neq (ext(B), ext(A))$

The most basic application...

Static Root of Trust Measurement (SRTM)

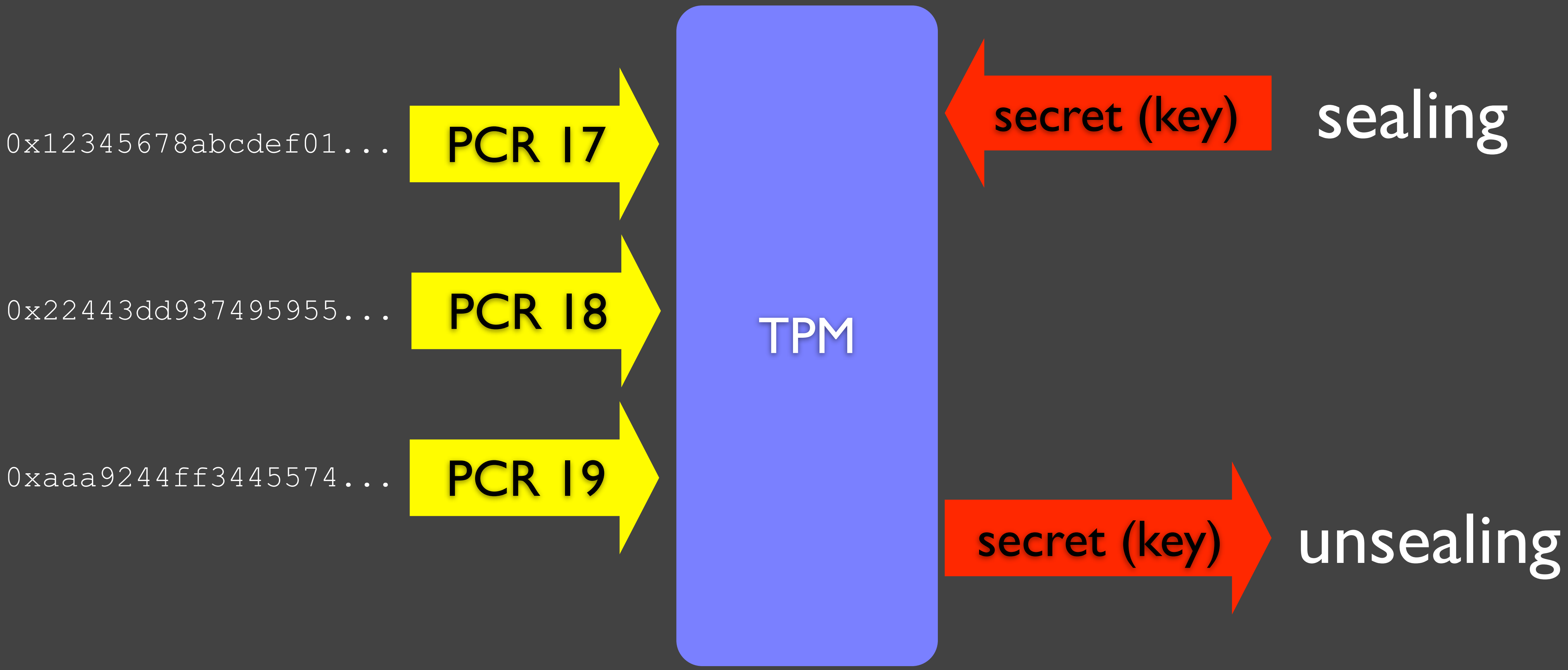


PCR	Usage (convention)
0	BIOS ROM & FLASH
1	Chipset config
2	PCI ROMs
3	PCI config
4	bootloader
5	bootloader config
6	...
7	...
8	e.g. OS kernel

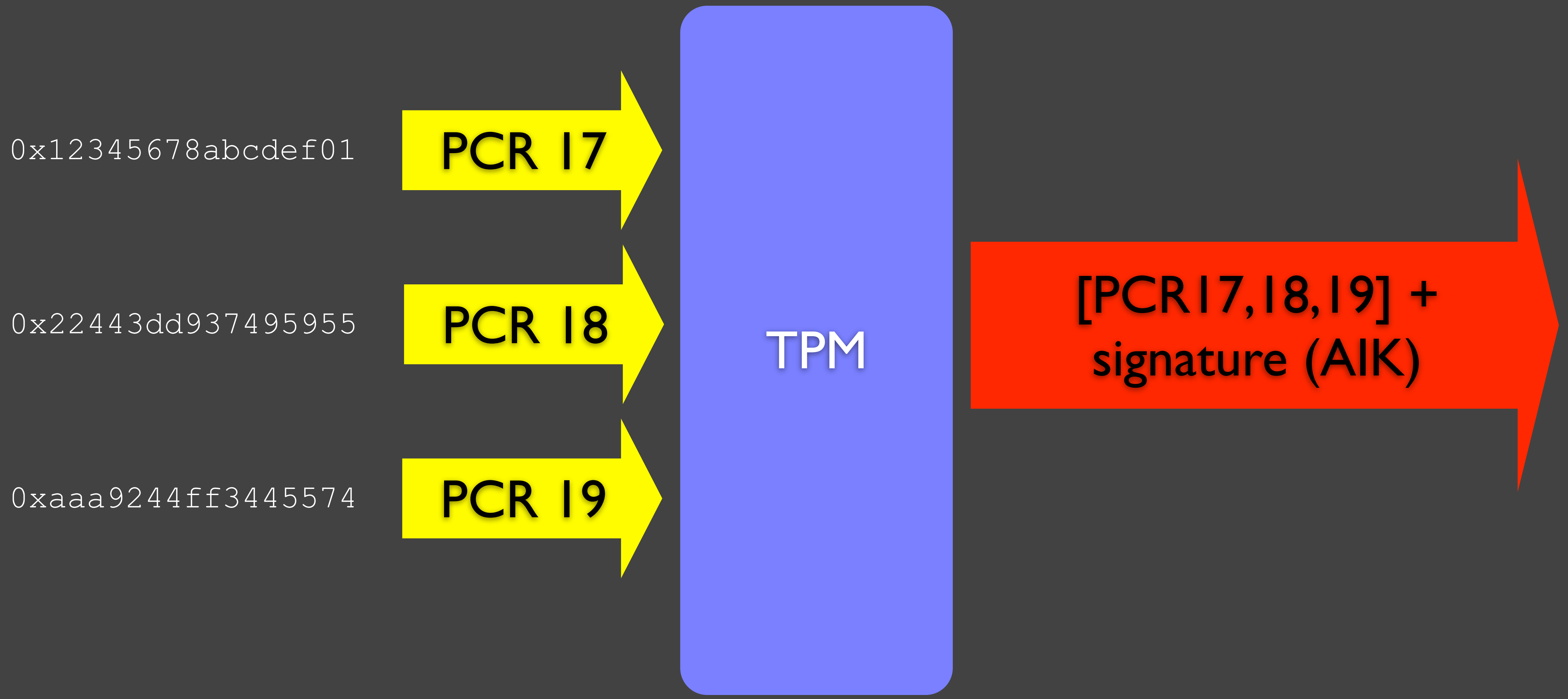
Why PCRs are so important?

Because of the Seal and Quote operations

TPM: Seal/Unseal Operation



TPM: Quote Operation (Remote Attestation)



TPM is passive!

It doesn't have a DMA engine -- cannot access host memory

Building Block #2: Intel Trusted Execution Technology (TXT)

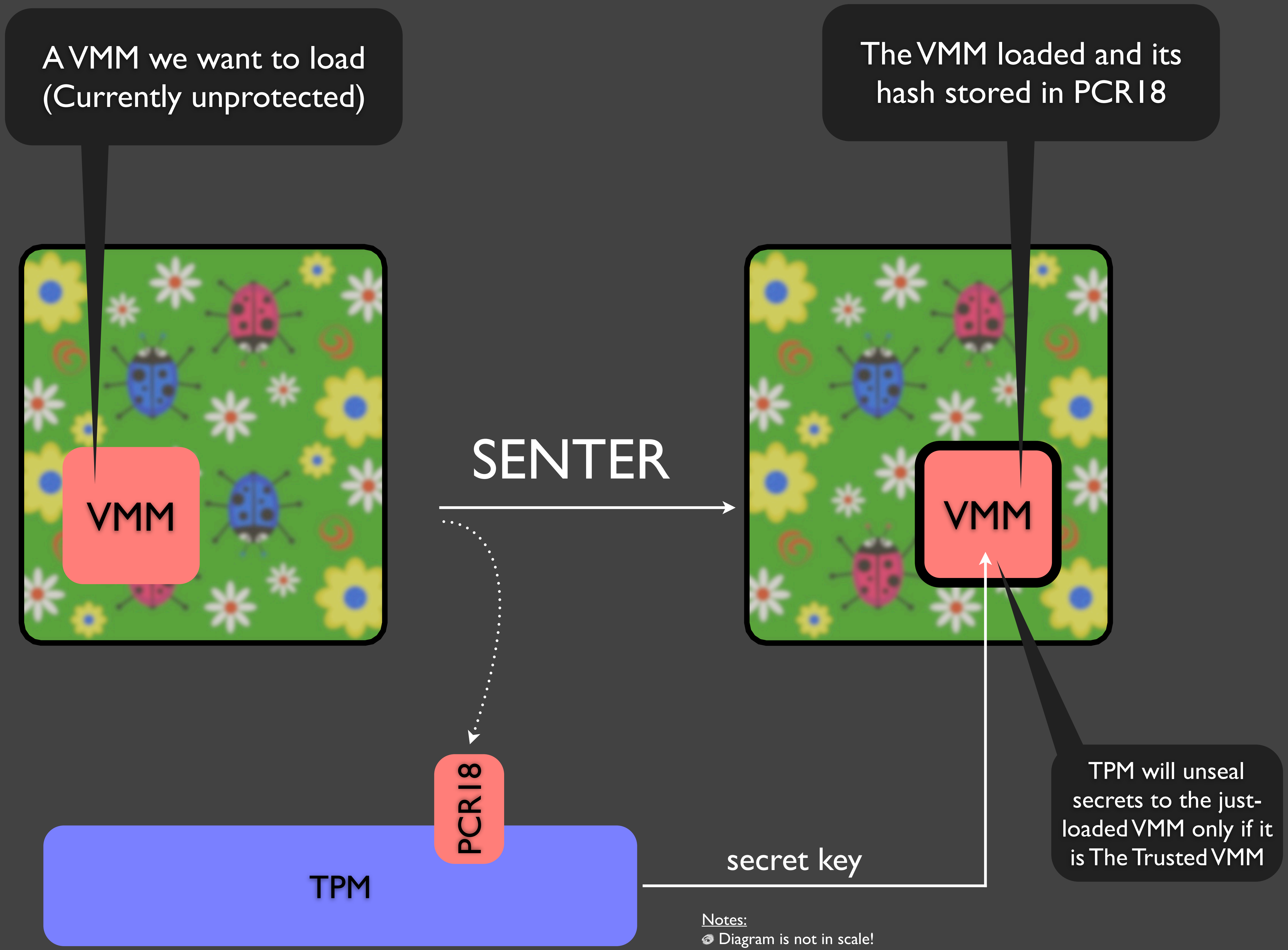
...AKA LaGrande

(renamed some 2 years ago)

Dynamic Root of Trust Measurement (DRTM)

SENTER — one of a few new instructions introduced by TXT

(They are all called SMX extensions)


A VMM we want to load
(Currently unprotected)
The VMM loaded and its hash stored in PCR18
VMM
SENTER
VMM
PCR18
TPM
secret key
TPM will unseal secrets to the just-loaded VMM only if it is The Trusted VMM

Notes:
- Diagram is not in scale!
- SENTER also resets and extends PCR17 with hash of SINIT/BIOSACM/(STM)/ LCP

- TXT late launch can transfer from unknown/untrusted/unmeasured system…
- to a known/trusted/measured system
- Without reboot!

- The system state ("trustedness") can be verified (possibly remotely) because all important components (hypervisor, kernel) hashes get stored into the TPM by SENTER.

SENTER will **not** block loading of untrusted VMM!

SENTER is not obligatory!!!

TXT and TPM: cannot enforce anything on our hardware! We can always choose *not* to execute SENTER!

So what is this all for?

Why would a user (or an attacker for that matter) be interested in executing the SENTER after all?

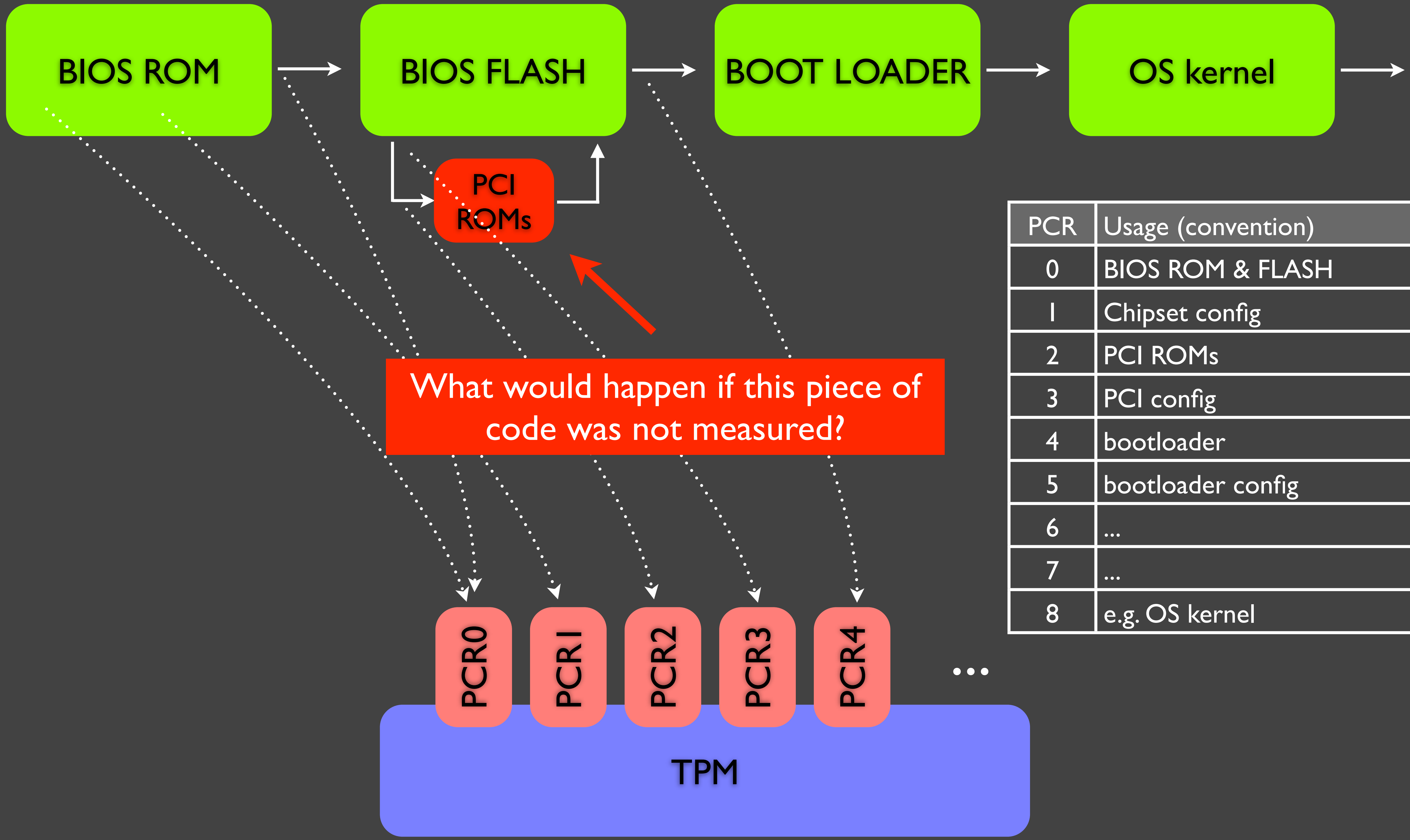
It’s all about TPM PCRs and secrets sealed in TPM!

(alternatively: about Remote Attestation)

SRTM vs. DRTM

Problems with SRTM

- COMPLETENESS — we need to measure *every* possible piece of code that might have been executed since the system boot!
- SCALABILITY of the above!



What would happen if this piece of code was not measured?

PCR	Usage (convention)
0	BIOS ROM & FLASH
1	Chipset config
2	PCI ROMs
3	PCI config
4	bootloader
5	bootloader config
6	...
7	...
8	e.g. OS kernel

For SRTM to make sense **all** possible code that might be executed should be measured and its hash stored in a PCR!

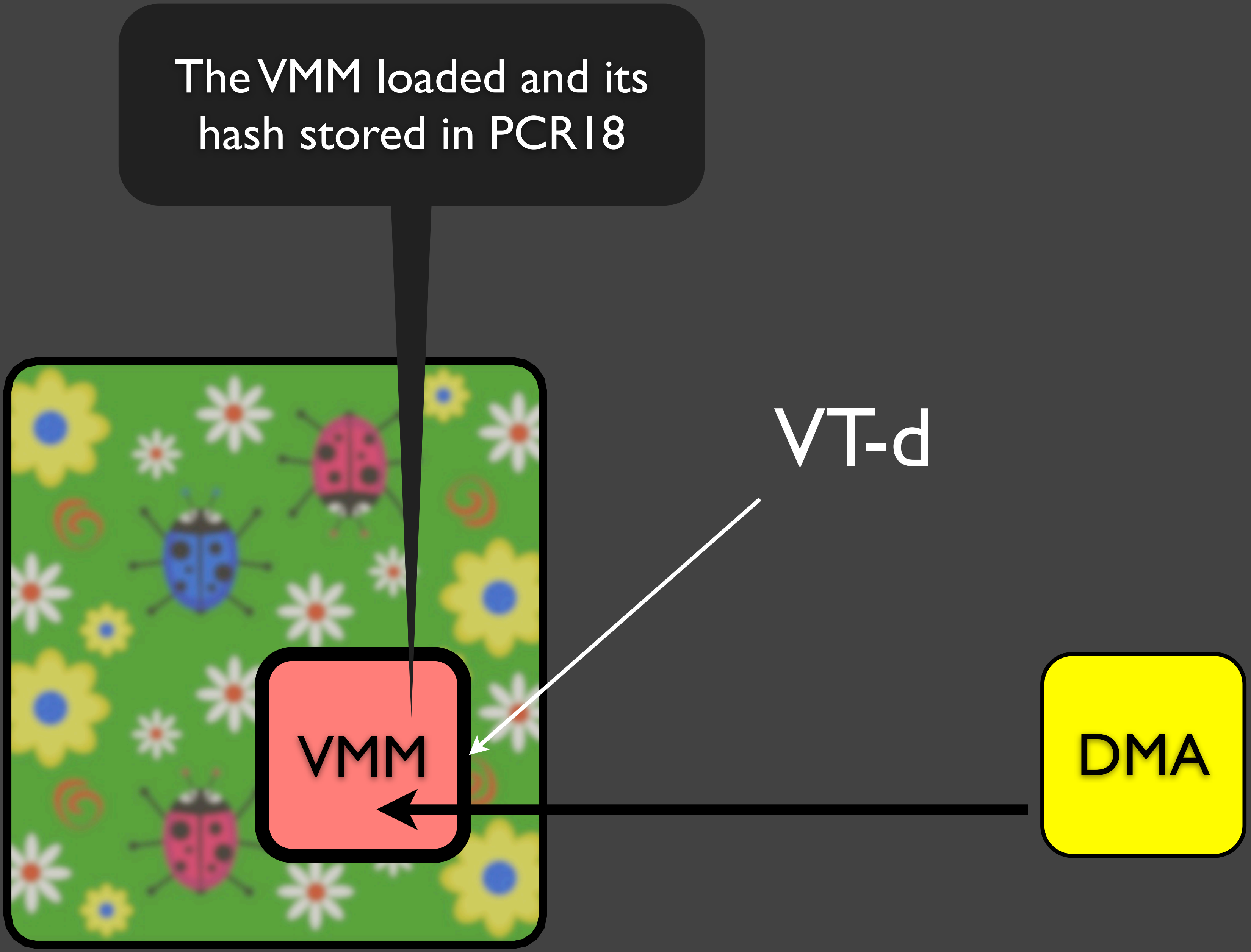
This might be hard in practice and this is why we have DRTM

Building Block #3: Intel Virtualization Technology (VT)

Intel VT

VT-x -- CPU virtualization
VT-d -- Device virtualization/remapping (IOMMU)

VT-d is crucial for TXT...



VT-d protects the VMM against malicious DMA attacks from PCI devices

TC technology on today's computers

Intel	**TPM**, TXT, VT-d
AMD	**TPM**, Presidio, IOMMU

Examples of Trusted Computing

#1: Evil Maid

So, you're a paranoid person and use disk encryption?

Feel secure against Laptop thieves?

Problem: The Evil Maid

Laptop **left alone** in a hotel room...

1. **Evil Maid** sneaks in and boots laptop from the **Evil USB**.
   - The USB **infects MBR** on your laptop (e.g. BluePillBoot)
   - Operating time: 3 minutes

2. User comes, boots the laptop, and **enters passphrase**...
   - BluePillBoot sniffs the decryption key and saves it somewhere, or transmits over the network...

3. Evil Maid can now steal the laptop -- she can decrypt it!

Have you ever left your laptop(s) unattended?

How TPM can help?

Trusted Boot

- Disk encrypted with a key k, that is sealed into the TPM...
- Now, only if the correct software (i.e. uninfected!) gets started it will get access to the key k and would be able to decrypt the disk!
- MS's Bitlocker works this way.

But...

- The Evil maid infects MBR that displays a fake password prompt (e.g. fake Bitlocker PIN screen)
- It stores the key on some unencrypted portion of disk (or send it via the network card) -- still before booting the OS...
- Th Evil Loader cannot hand down execution to Bitlocker -- it would not boot the system. So, it pretends the password/PIN entered was wrong and reboots the system (but first it restores the MBR to the original one)

A really paranoid user should thus **destroy** his/her laptop, if entered correct password, but the OS didn't boot!

or...

User's Picture Test

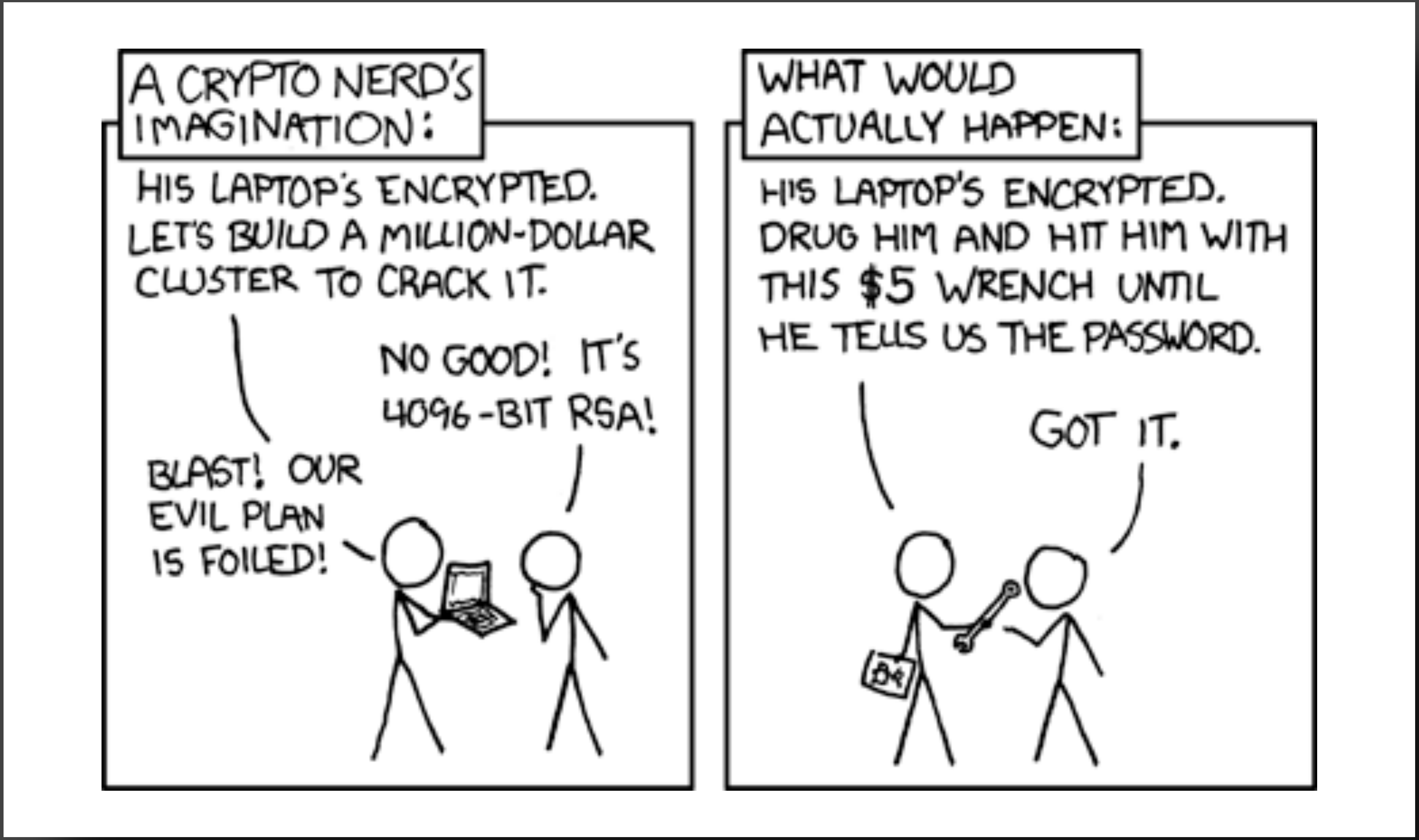
- During installation, a user takes a picture of themselves using a built-in in laptop camera...
- This picture is stored on disk, encrypted with key k_{pic}, which is sealed by the TPM…
- Now, on each reboot — only if the correct software got loaded, it will be able to retrieve the key k_{pic} and present a correct picture to the user. Only then the user enters the passphrase!
- Important: after the use accepts the picture, the software should extend PCR's with some value (e.g. 0x0), to lock access to the key k_{pic}

This way the Evil Maid cannot prepare a fake Bitlocker prompt!

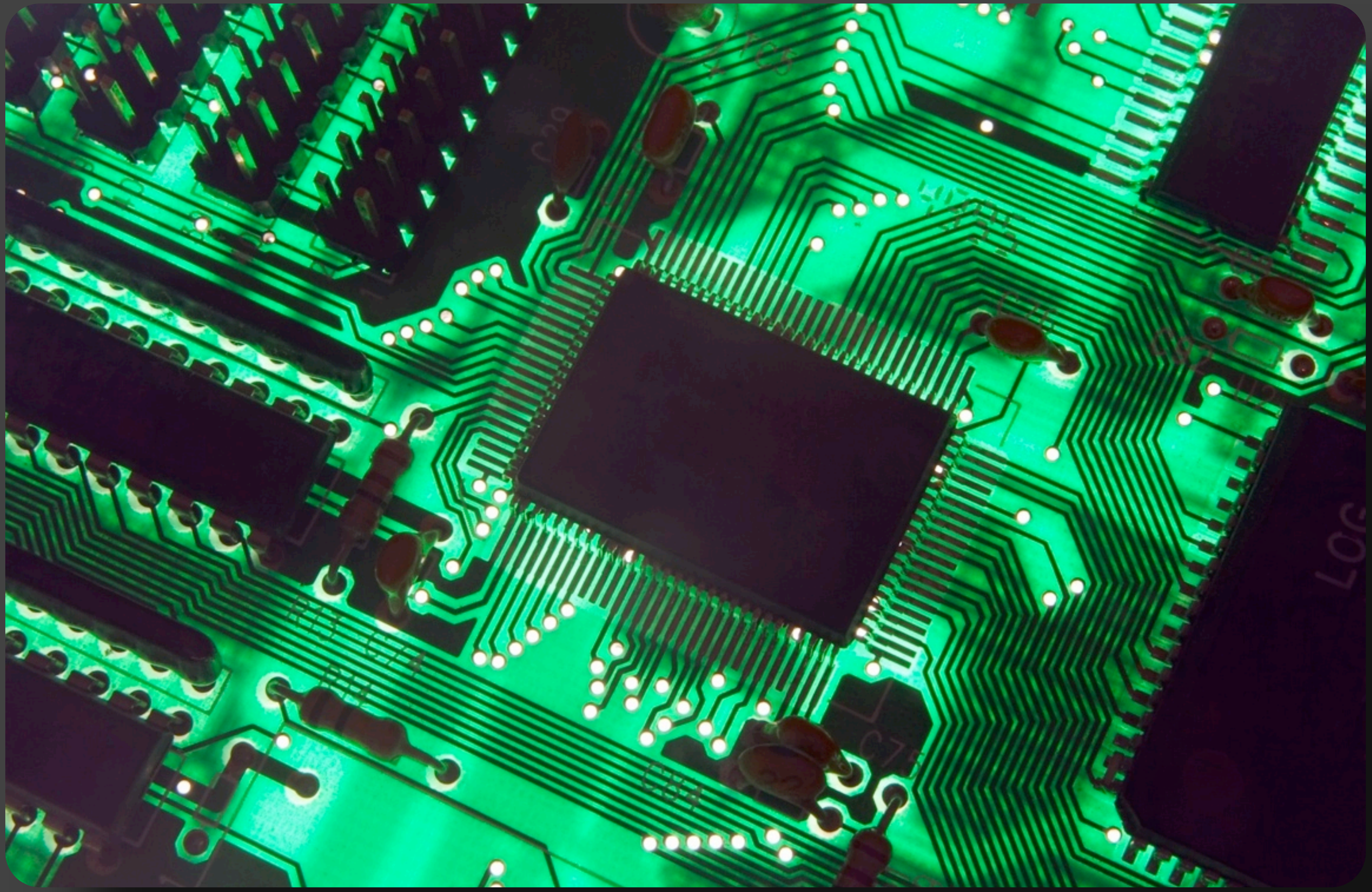
No Big Deal!

- h/w keyboard sniffer
- hidden camera
- e/m leak

} solution: keyfiles, tokens, etc


A CRYPTO NERD'S IMAGINATION:
HIS LAPTOP'S ENCRYPTED. LET'S BUILD A MILLION-DOLLAR CLUSTER TO CRACK IT.
NO GOOD! IT'S 4096-BIT RSA!
BLAST! OUR EVIL PLAN IS FOILED!
WHAT WOULD ACTUALLY HAPPEN:
HIS LAPTOP'S ENCRYPTED. DRUG HIM AND HIT HIM WITH THIS $5 WRENCH UNTIL HE TELLS US THE PASSWORD.
GOT IT.

source: http://xkcd.com/



#2: "Chinese" Hardware Backdoors

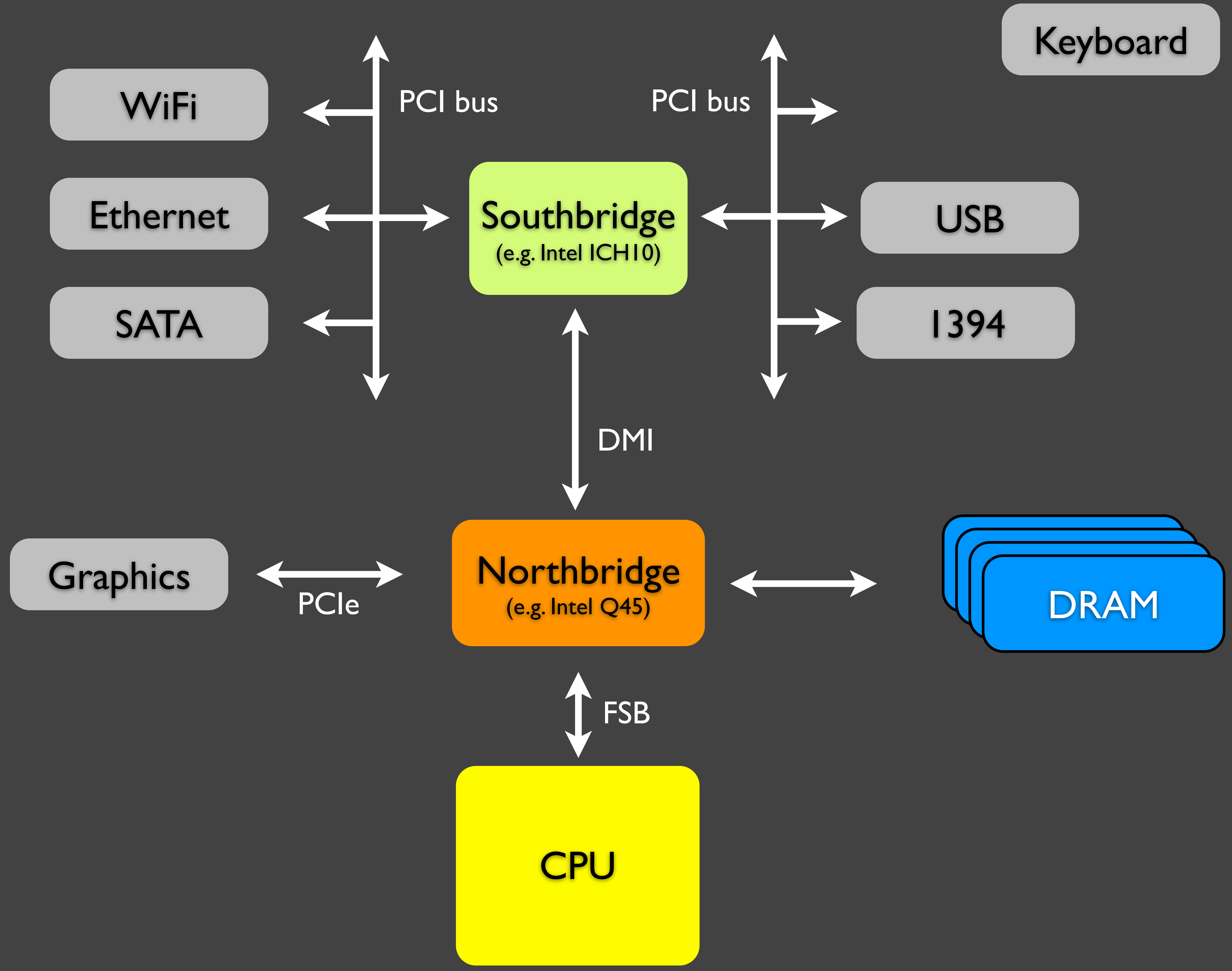
How many laptops/parts are made in China*?

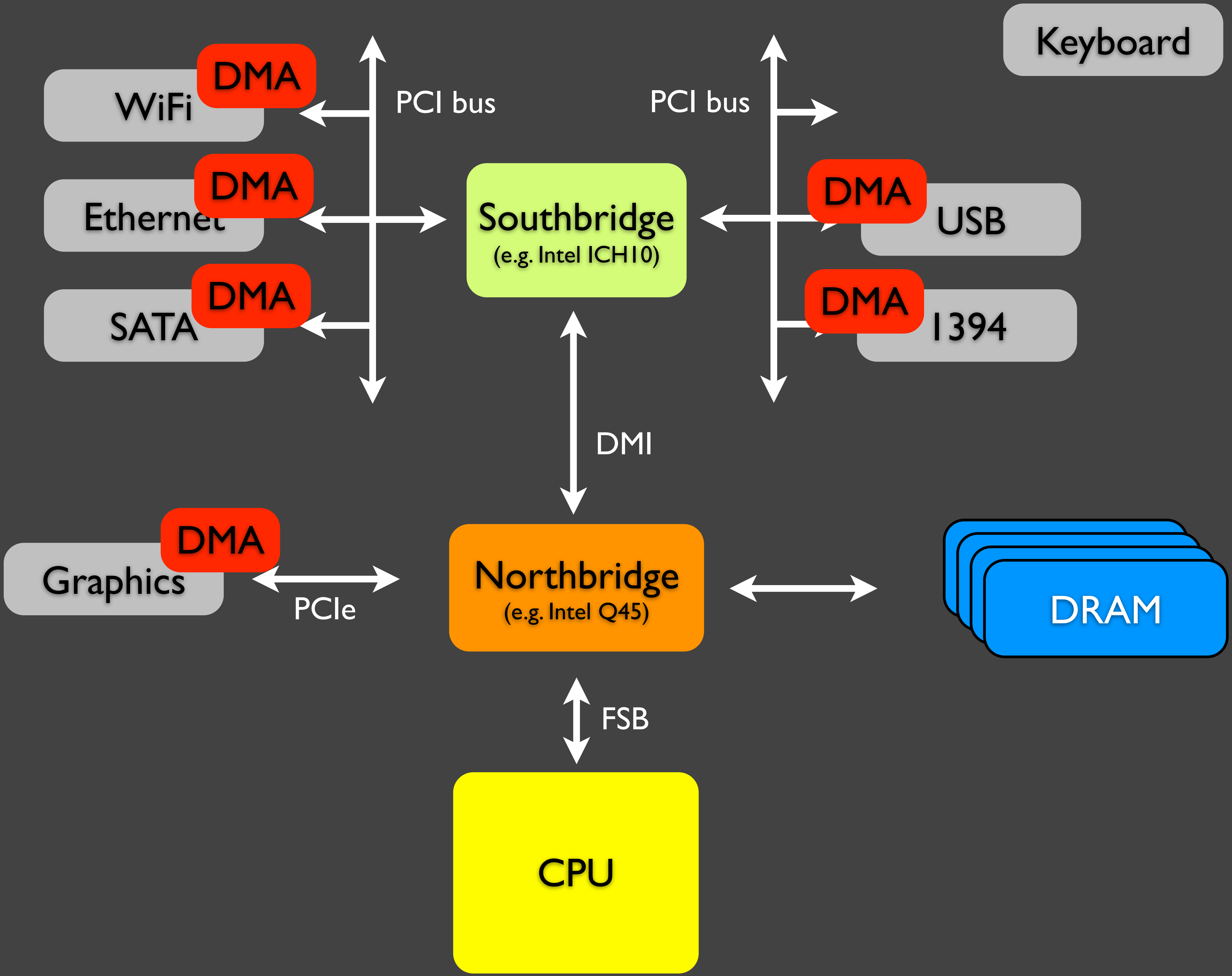
*Substitute with your favorite evil country

Afraid of malicious vendors?

A "Made in China"* PCI-based backdoor?

*Substitute with your favorite evil country


Keyboard
WiFi
PCI bus
PCI bus
Ethernet
Southbridge
(e.g. Intel ICH10)
USB
SATA
1394
DMI
Graphics
PCIe
Northbridge
(e.g. Intel Q45)
DRAM
FSB
CPU

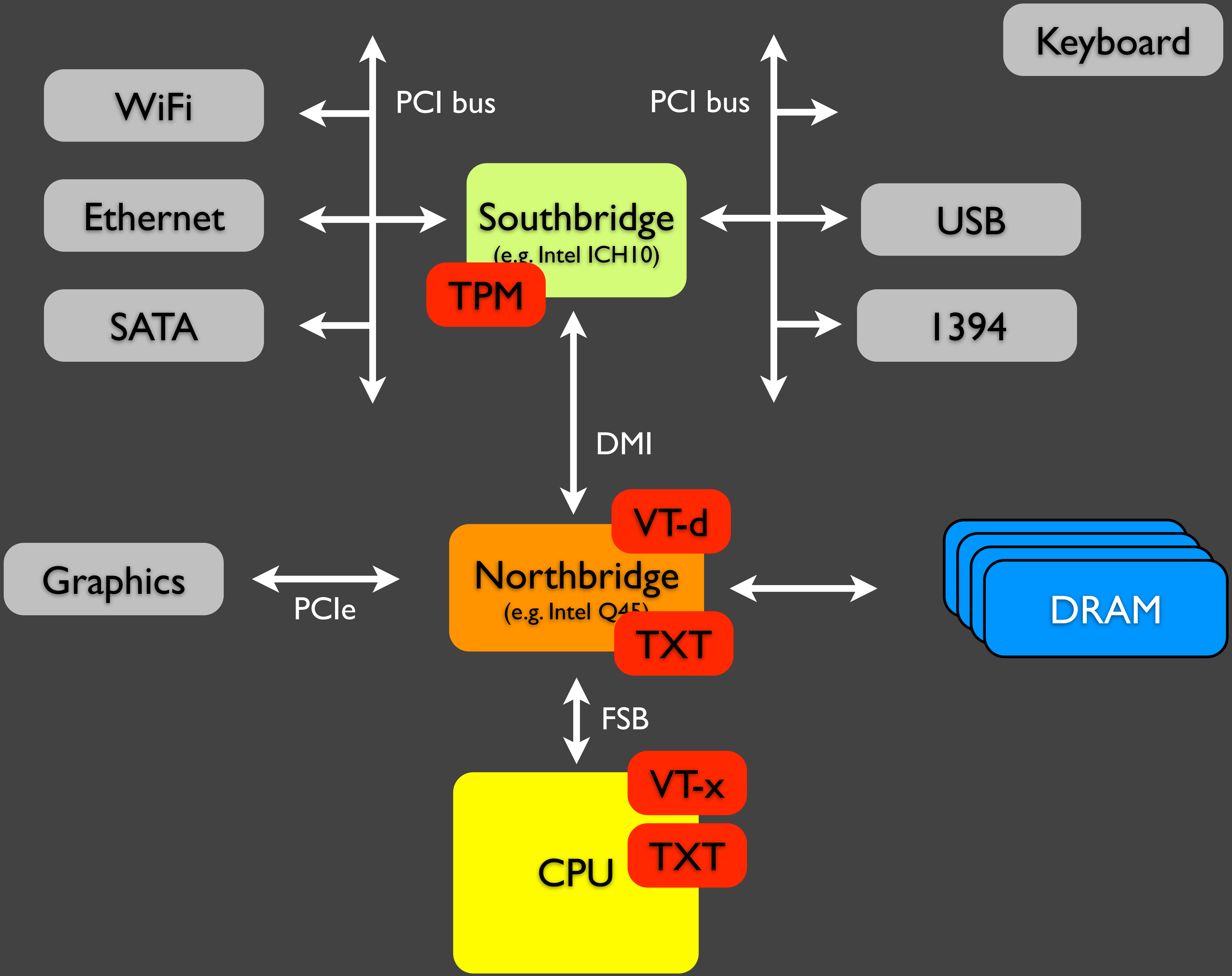

Keyboard
DMA
WiFi
PCI bus
PCI bus
DMA
Ethernet
Southbridge
(e.g. Intel ICH10)
DMA
USB
DMA
SATA
DMA
1394
DMI
DMA
Graphics
PCIe
Northbridge
(e.g. Intel Q45)
DRAM
FSB
CPU

Each device that has a DMA engine can access the **whole system memory**! Including kernel!

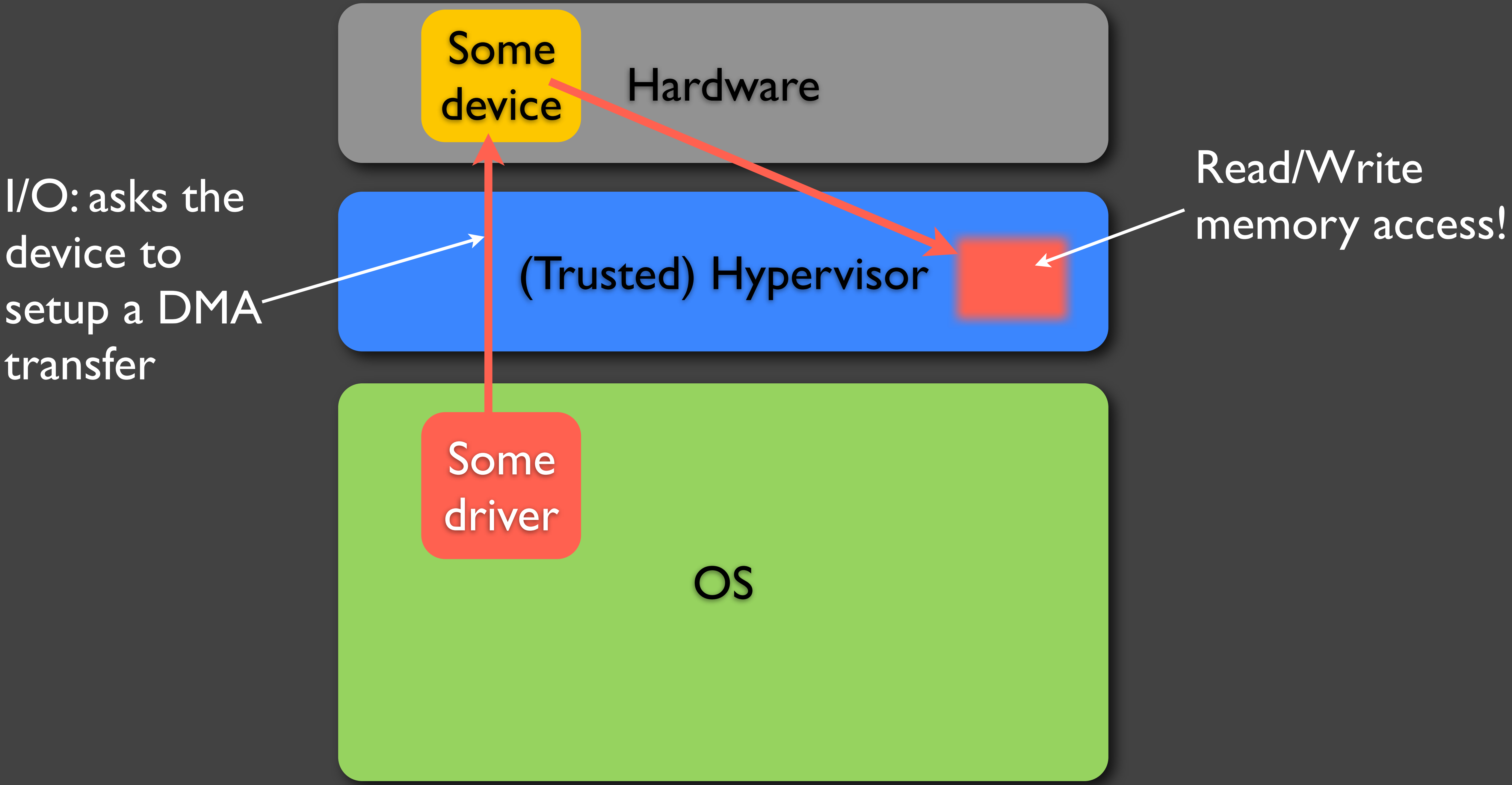
Southbridges (e.g. Intel ICHx) have many PCI devices integrated, e.g.:

- SATA controllers
- USB controllers/hubs
- 1394/Firewire controllers
- Ethernet cards
- etc...

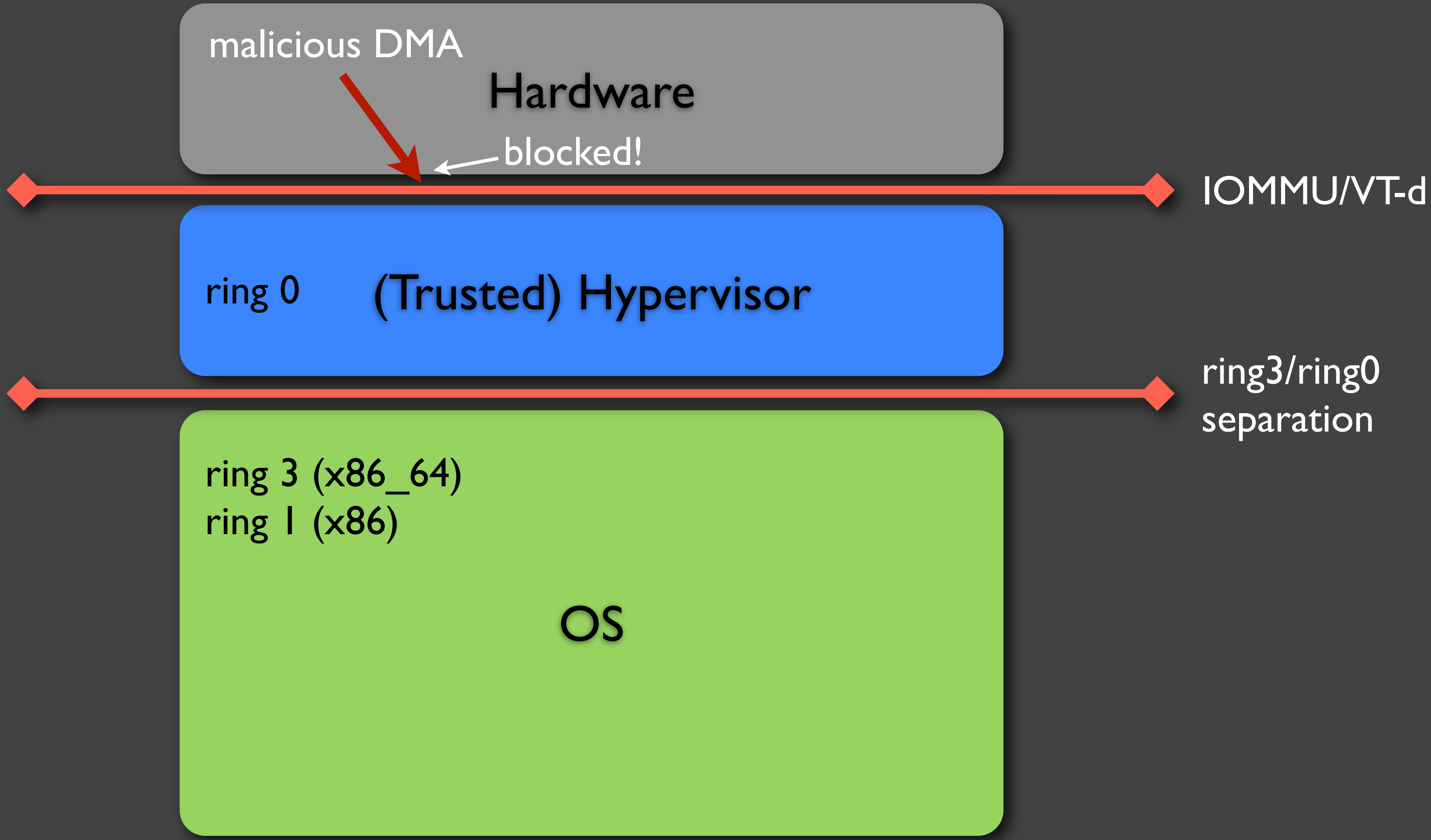
How TC comes into play?


Keyboard
WiFi
PCI bus
PCI bus
Ethernet
Southbridge
(e.g. Intel ICH10)
TPM
USB
SATA
1394
DMI
VT-d
Graphics
Northbridge
(e.g. Intel Q45)
TXT
DRAM
PCIe
FSB
VT-x
CPU
TXT

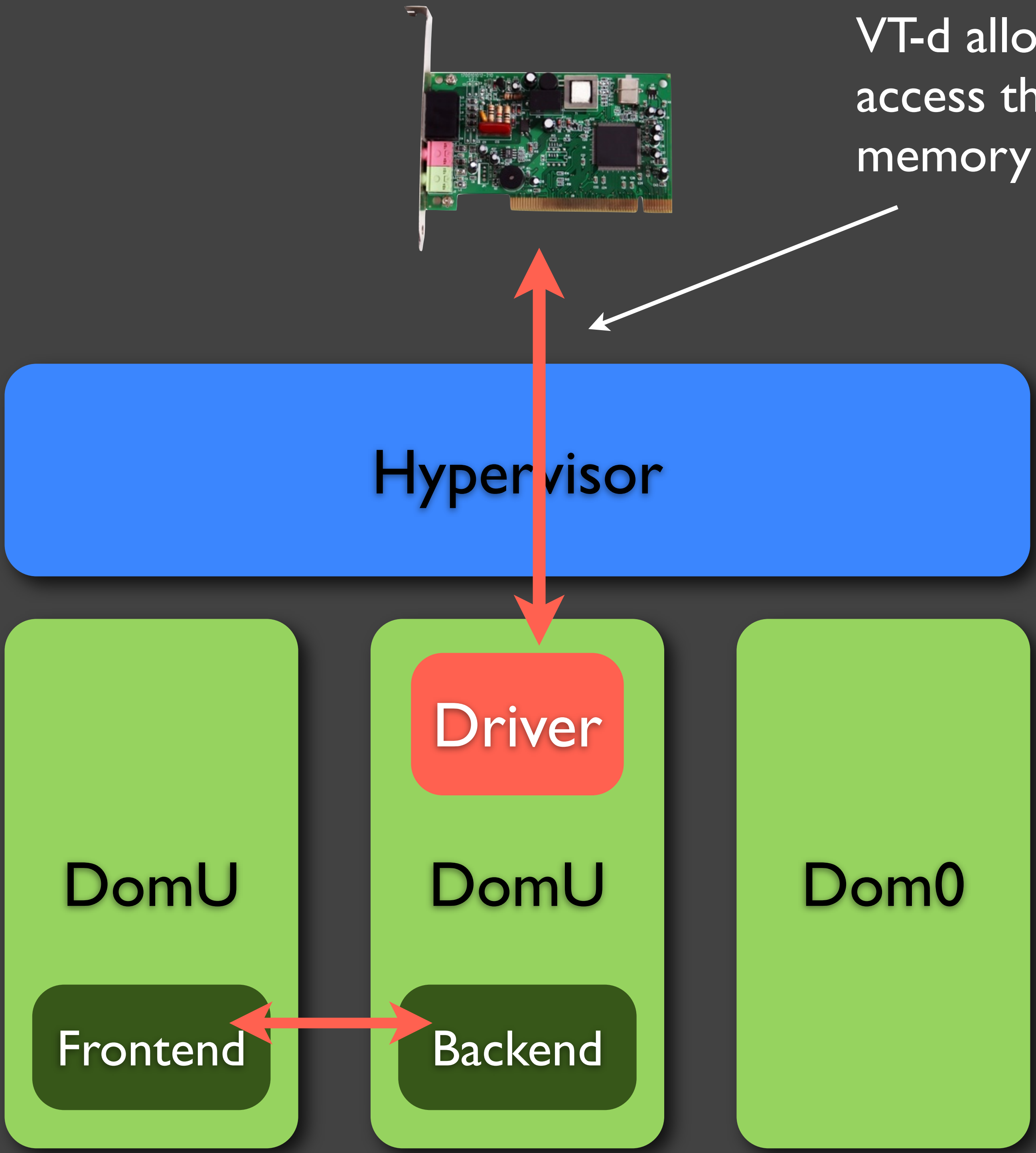
VT-d can block unwanted DMA from devices


Some device
Hardware
I/O: asks the device to setup a DMA transfer
(Trusted) Hypervisor
Read/Write memory access!
Some driver
OS

Xen and VT-d

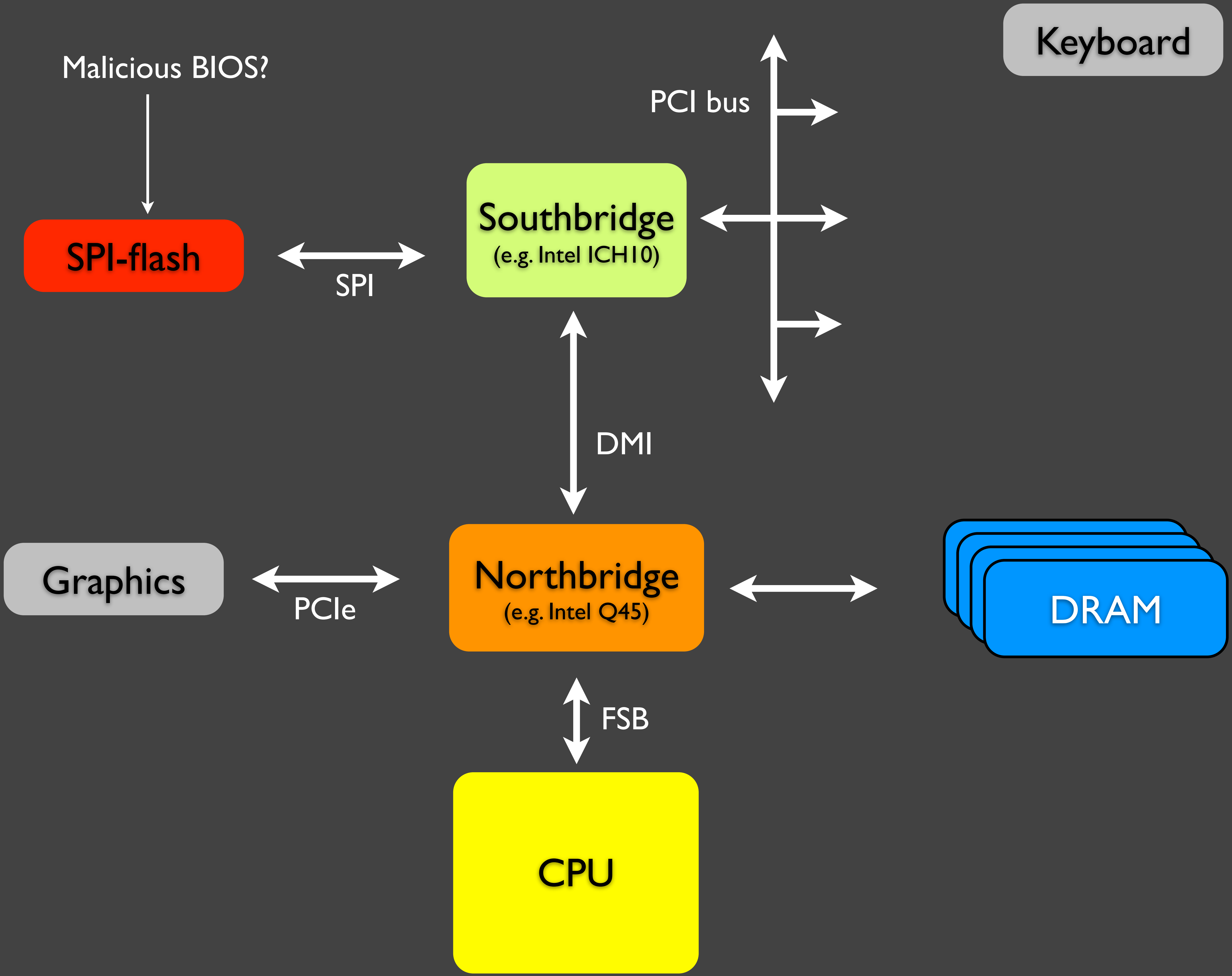

malicious DMA
Hardware
blocked!
IOMMU/VT-d
ring 0
(Trusted) Hypervisor
ring3/ring0
separation
ring 3 (x86_64)
ring 1 (x86)
OS

VT-d can be programmed to allow DMA from devices only to limited memory addresses (e.g. occupied by specific driver(s))



VT-d allows this NIC only to access this driver domain's memory

Malicious firmware?


Keyboard
Malicious BIOS?
PCI bus
SPI-flash
SPI
Southbridge
(e.g. Intel ICH10)
DMI
Graphics
PCIe
Northbridge
(e.g. Intel Q45)
DRAM
FSB
CPU

We shall fear it not!

TXT for the rescue!
With DRTM we do not need to trust the BIOS!
We do not need to maintain a chain of trust starting from CRTM!

Conclusion: VT-d and TXT should be able to protect us against malicious hardware backdoors!

Let's repeat it, as it is important conclusion...

Conclusion: VT-d and TXT should be able to protect us against malicious hardware backdoors!

...except...

...except for the backdoors in the chipset or CPU!

Shall we trust Intel or AMD?

Building in a backdoor into a processor is trivial

A local priv-escalation "enabler":

```
if (rax == MAGIC_1 && rcx == MAGIC_2) {cpl=0; jmp [rbx];}
```

... only a few more gates ;)

How many people can reverse engineer a processor?

But TC does **not** make it any easier!
It's totally irrelevant whether TC is present or not.

...yet at the same time TC protects against many other possible hardware backdoors.

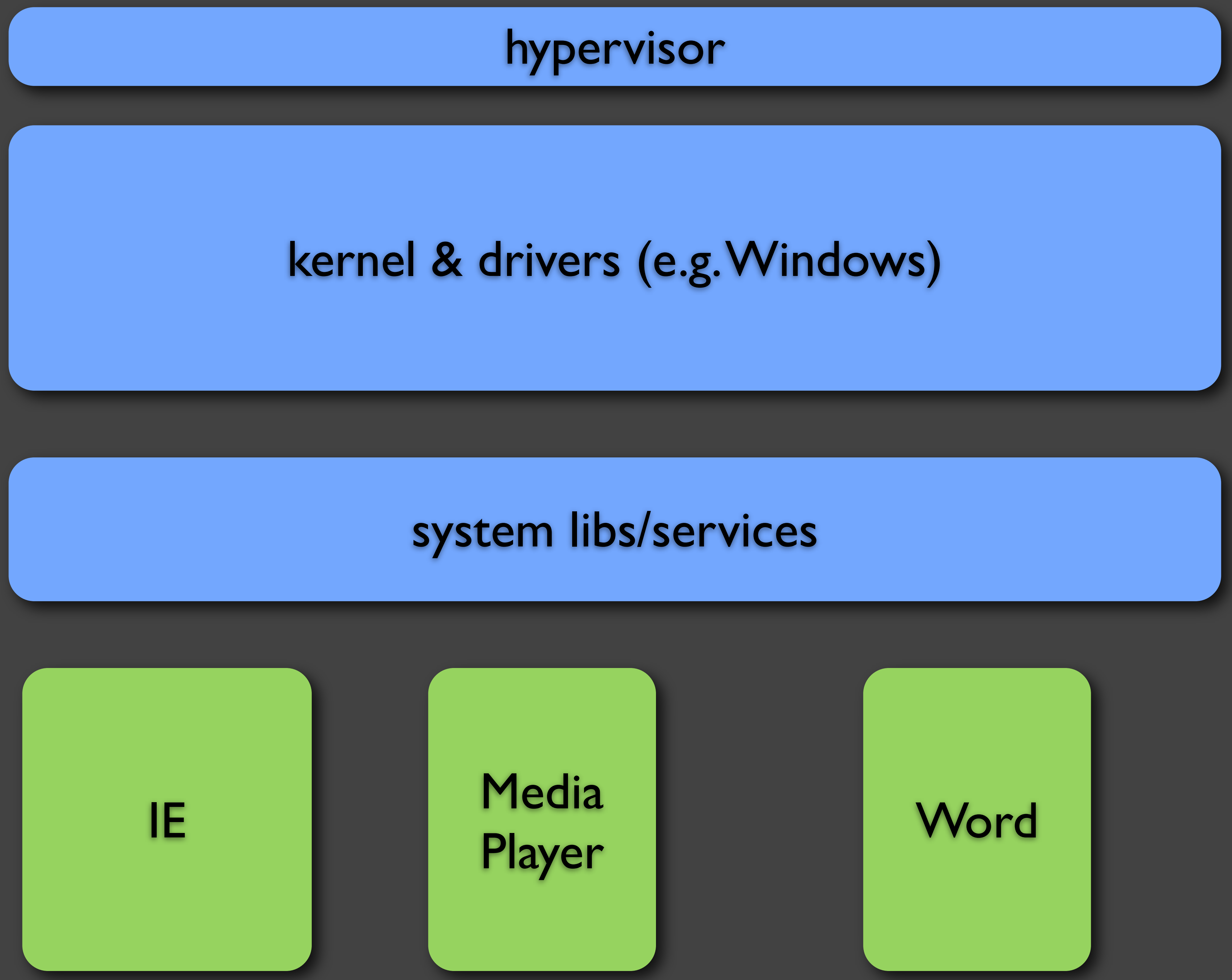
#3: Evil DRM?

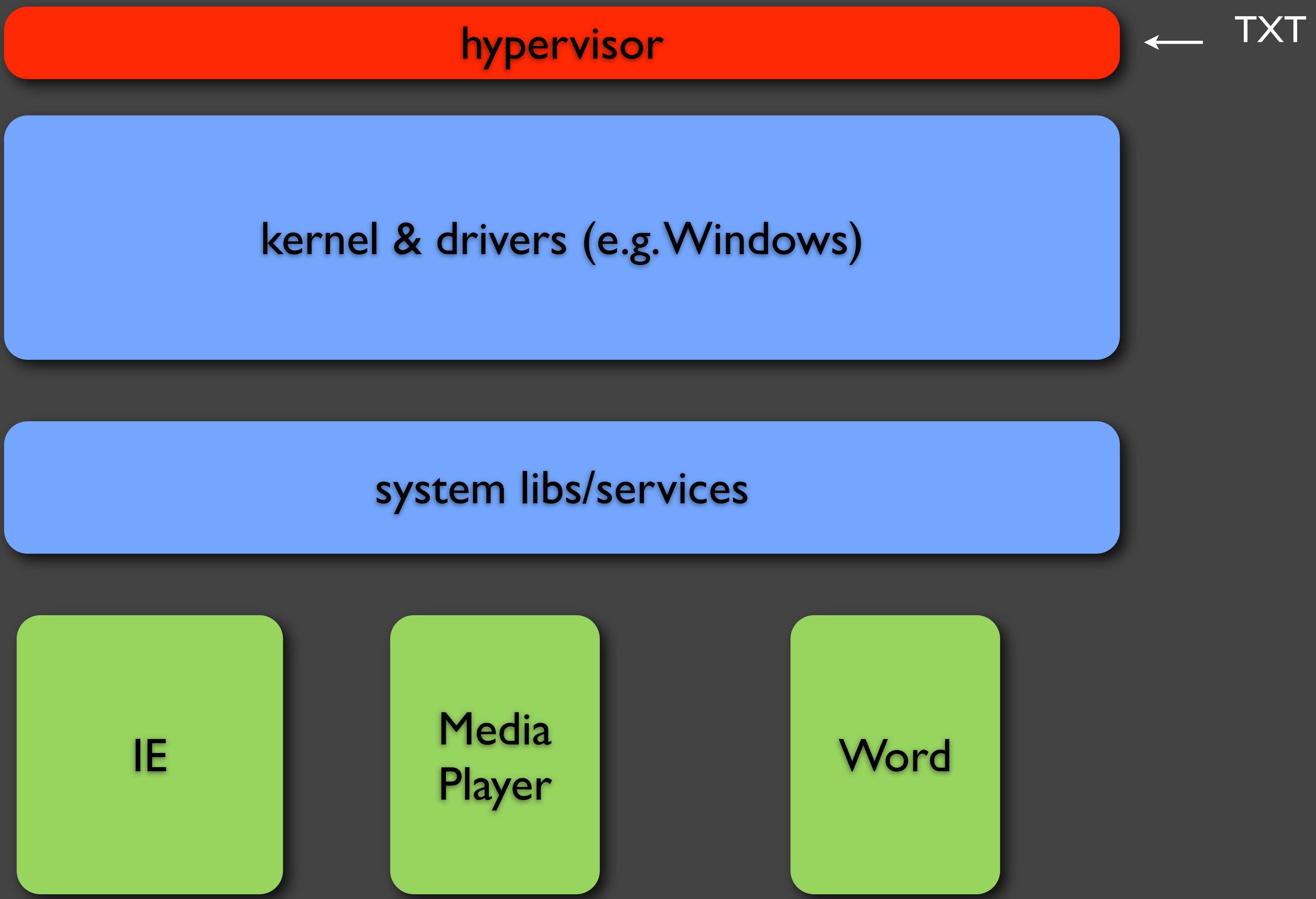
You don't use TXT to load e.g. `tetris.exe`

(or `mediaplayer.exe` for that matter)

You use TXT to load a VMM (hypervisor)!

in other words you use it to load **the whole system!**


hypervisor
kernel & drivers (e.g. Windows)
system libs/services
IE
Media Player
Word


hypervisor
TXT
kernel & drivers (e.g. Windows)
system libs/services
IE
Media Player
Word

1. VMM securely loads the kernel,
2. Kernel securely loads the drivers,
3. Kernel securely loads libs/services,
4. Kernel securely loads critical apps (e.g. media player).

Sounds good?

But what about runtime compromises?

1. VMM should make sure kernel cannot be compromised!
2. Kernel/VMM should make sure drivers cannot be compromised!
3. Kernel should make sure libs/services cannot be compromised!
4. Kernel should make sure securely loads critical apps (e.g. media player) cannot be compromised!

Here we protect against compromises <u>at runtime!</u>

Protecting against driver compromise at runtime is possible if drivers were properly isolated and IOMMU was used.

But this means a total redesign of Windows architecture!
Effectively, a migration towards microkernel-based OS!

Can you imagine MS doing this anytime soon?

Protection against runtime application compromise is simply infeasible today and in the near future!

(People has been announcing the end of buffer overflows for nearly a decade -- but no visible progress in practice)

TXT as a protection of your core OS components: **yes**
TXT as a protection of all the Apps: **no!**

Example #1: Evil Maid
Example #2: Chinese backdoors
Example #3: DRM

There are more...

Theory vs. Reality

Attacking Trusted Computing

Hardware-based Attacks

Requires physical access to the machine;
Cannot be used by malware

Software-only Attacks

Ideal for malware

Hardware Attacks

Hardware-based attacks: not such a big deal, really!

- TPM reset attacks using a metal clip
- LPC bus interceptions
- Reading TPM nv-storage using microscope
- etc.

but...

TPM getting integrated into chipsets

e.g. Intel ICH10 has integrated TPM

Now attacker needs to intercept DMI bus (2GB/s)
+ the communication might(*) be encrypted

(*) We haven't checked if this channel is encrypted on Q45/ICH10 chipset. But could be.

But even a successful physical attack on TPM (that might even result in obtaining all the keys), doesn't automatically allow to e.g. bypass user disk encryption (e.g. Bitlocker)

TPM is only needed to provide trusted boot -- the booted application is still required to obtain the password from the user

So, a successful attack on TPM could, at best, allow for a successful Evil Maid attack that we discussed before.

(Without TPM this attack is always possible)

Software Attacks

Attacks against SRTM

OSLO: Improving the security of Trusted Computing

by Bernhard Kauer, 2007

Kauer's attacks:

1. Buggy **bootloaders** that do not maintaing complete chain of trust
2. TPM 1.1 software **reset**
3. Intercept some BIOS'es **CRTM**

This was in 2007 and most problems fixed now.

Also, the attacks didn't affect the core technologies

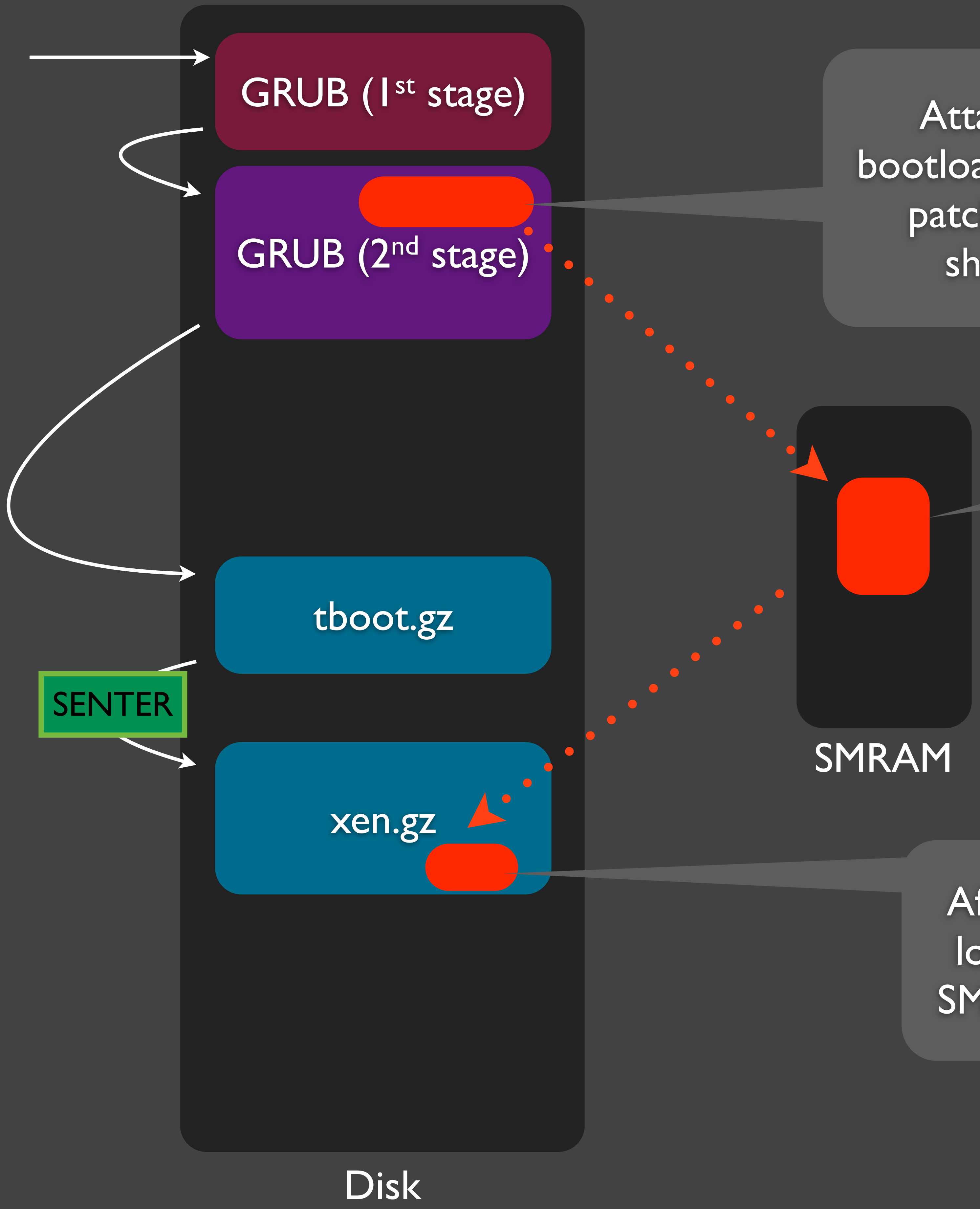
(TPM attack was against 1.1, not 1.2)

Attacks against DRTM (TXT)

Attacking Intel TXT

by ITL, Feb 2009, Black Hat DC

TXT attack sketch (using tboot+Xen as example)



Attacker patches the bootloader (e.g. GRUB). The patched code injects a shellcode to SMM

Evil shellcode will infect the Xen hypervisor later...

After xen.gz gets sucesfully loaded, the evil code from SMRAM can easily infect it...

Notes:
- Diagram is not in scale!
- SENTER also resets and extends PCR17 with hash of SINIT/BIOSACM/(STM)/ LCP

~ February 2009

Intel

Invisible Things Lab

Intel: Solution to the TXT attack is called: STM

Invisible Things Lab: Can we take a look at this STM?

Intel: STM is currently not available.

Invisible Things Lab: ?

Intel: It is simple to write. There was just no market demand yet.

Invisible Things Lab: ?

The dialogs between ITL and Intel presented here have been modified for brevity and for better dramatic effect.

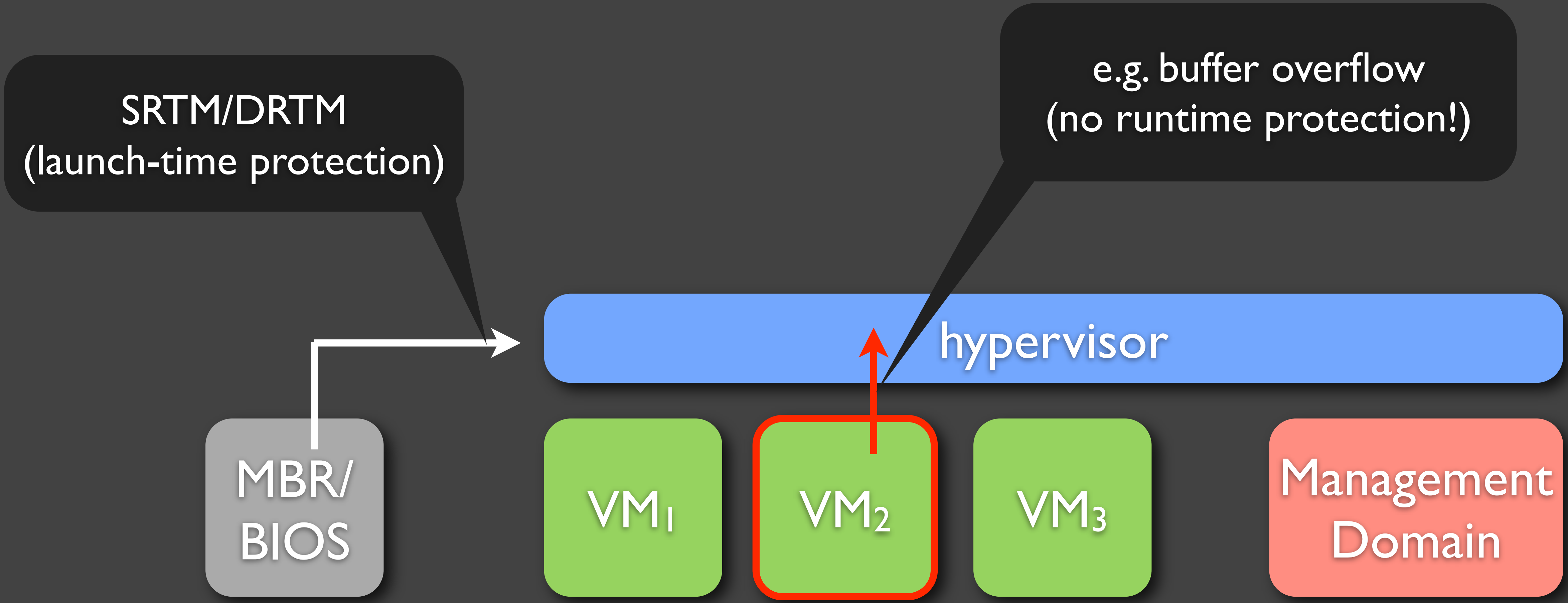
Is there any STM out there today?

I haven't heard of one yet...

TXT is bypassable on systems that do not have STM

(well, on most (all?) systems today)

Launch time protection vs. runtime protection


SRTM/DRTM
(launch-time protection)
e.g. buffer overflow
(no runtime protection!)
hypervisor
MBR/
BIOS
VM1
VM2
VM3
Management
Domain

By definition TXT/TPM cannot solve the problem of runtime attacks, such as buffer overflows.

VT-d can help though (by isolating drivers)

Think about TC as of a way to provide **trusted boot**
(launch-time protection)

...even that would be a big deal though

Final Thoughts

You shall not fear TPM and TXT

TC has potential to rise the bar for desktop security

(assuming the software will properly exploit them)

TPM, VT, and TXT are cool!

... as is the challenge of breaking them ;)

New Stuff Coming Soon...

This Summer

http://invisiblethingslab.com

ITL

INVISIBLE THINGS LAB