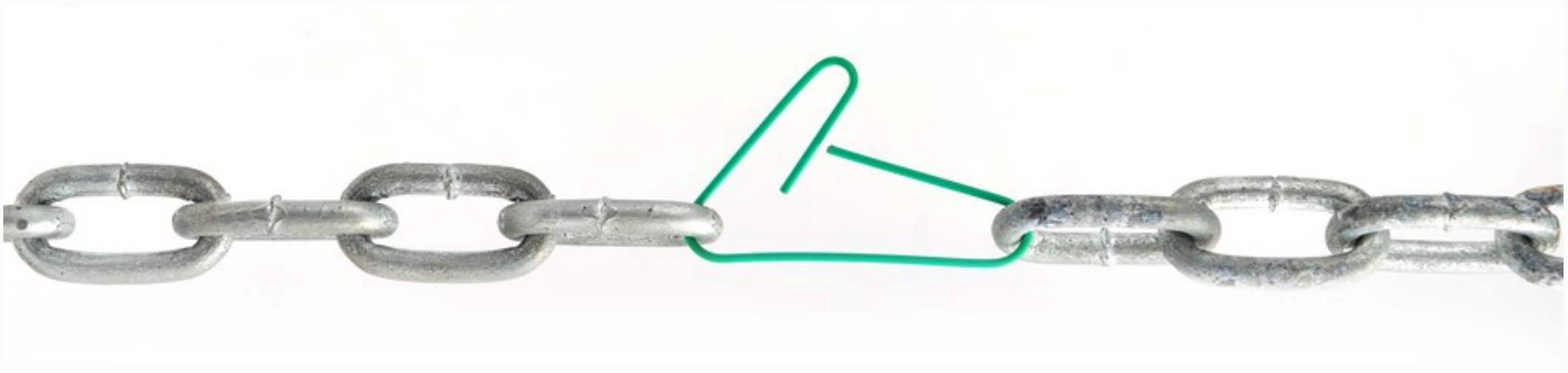
A Password is Not Enough

Why disk encryption is broken and how we might fix it

Daniel Selifonov

DEF CON 21
2013-08-02



Five Questions

- Do you encrypt the drive in your computer?
- Using something like TrueCrypt, dmcrypt, loop-aes?
- Do you always cold shutdown when leaving your computer unattended?
- Have you ever left your computer unattended for more than a few hours?
- How about more than a few minutes?

Why do we encrypt?

- Confidentiality and Integrity
- Secrecy, privacy, and the power to determine what happens to personal and business data
- Legal liability
- Access control in the face of physical access
- Trustworthiness of our endpoints

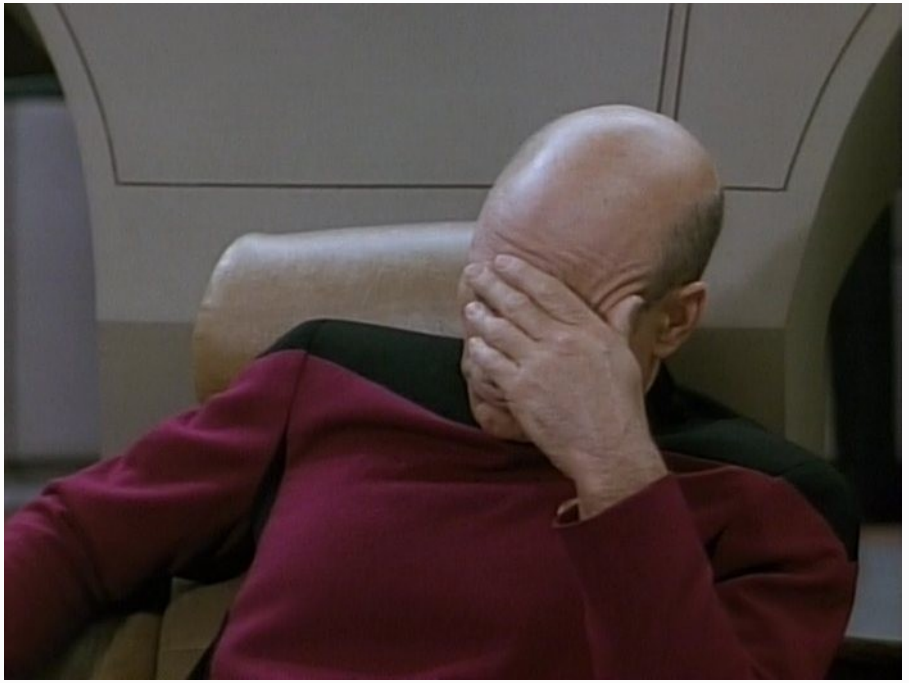
Armor, Sieves, and Rubber Hoses

- Cryptography is not the weakest link
  - Random number generation
  - Block cipher modes of operation
  - Key derivation from passwords
- Many open source implementations
- The side channels:
  - Attack the apparatus
  - Attack the user

http://en.wikipedia.org/wiki/File:Bush_mission_accomplished.jpg

Mismatched Objectives

- Disk encryption threat models versus de facto use:
  - “[TrueCrypt does **not**] secure any data on a computer if an attacker has physical access to the computer before or while TrueCrypt is running on it.”



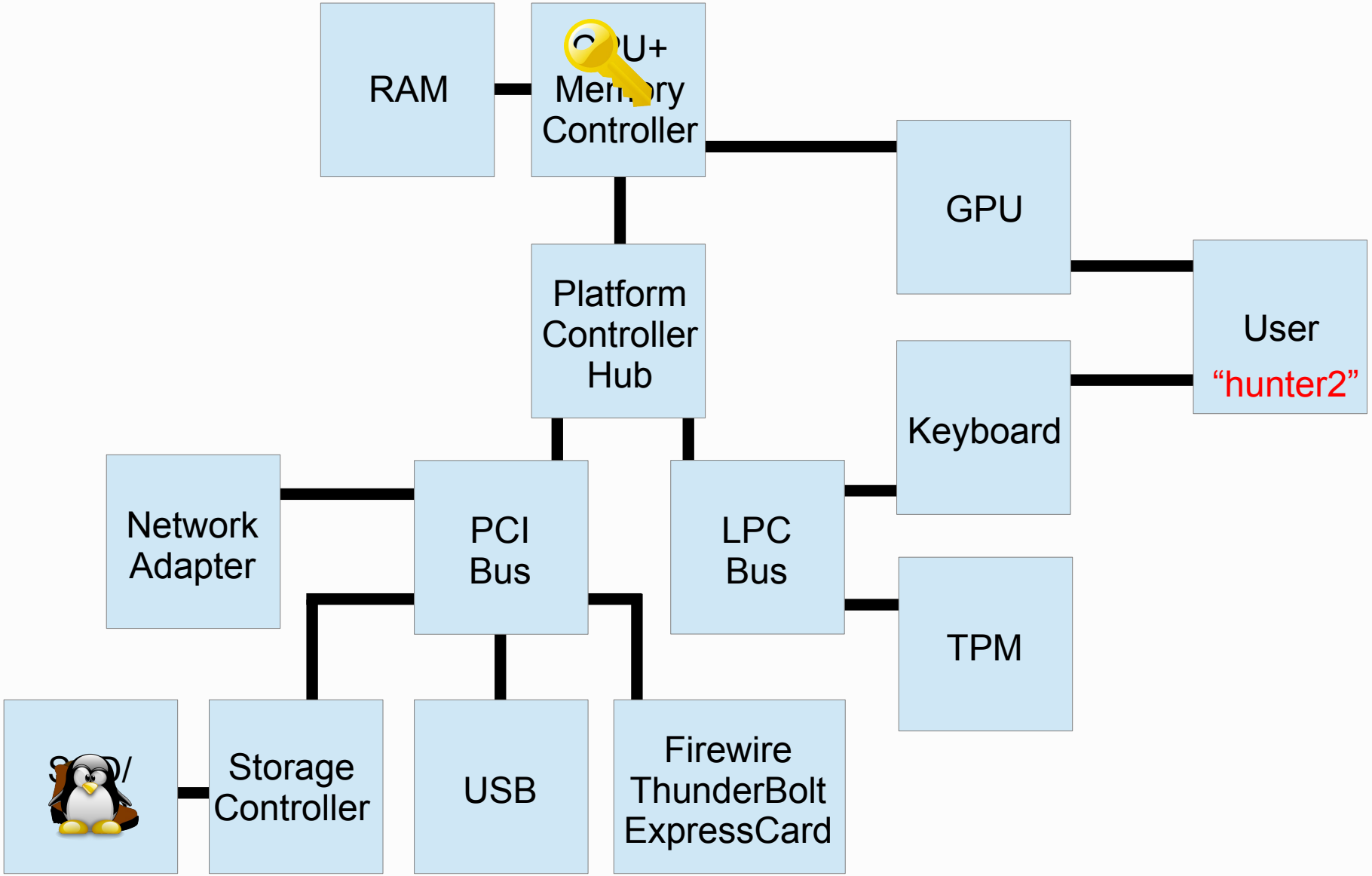
http://www.truecrypt.org/docs/security-model

From the Horse's Mouth

- TrueCrypt: We generally disregard "janitor" attacks since they inherently make the machine untrusted. We never consider the feasibility of hardware attacks; we simply have to assume the worst. After an attacker has "worked" with your hardware, you have to stop using it for sensitive data. It is impossible for TPM to prevent hardware attacks (for example, using hardware key loggers, which are readily available to average Joe users in computer shops, etc.)
- Joanna Rutkowska: And how can you determine that the attacker have or have not "worked" with your hardware? Do you carry your laptop with you all the time?
- TC: Given the scope of our product, how the user ensures physical security is not our problem. Anyway, to answer your question (as a side note), you could use e.g. a proper safety case with a proper lock (or, when you cannot have it with you, store it in a good strongbox).
- JR: If I could arrange for a proper lock or an impenetrable strongbox, then why in the world should I need encryption?

http://theinvisiblethings.blogspot.com/2009/10/evil-maid-goes-after-truecrypt.html

FDE Boot Process



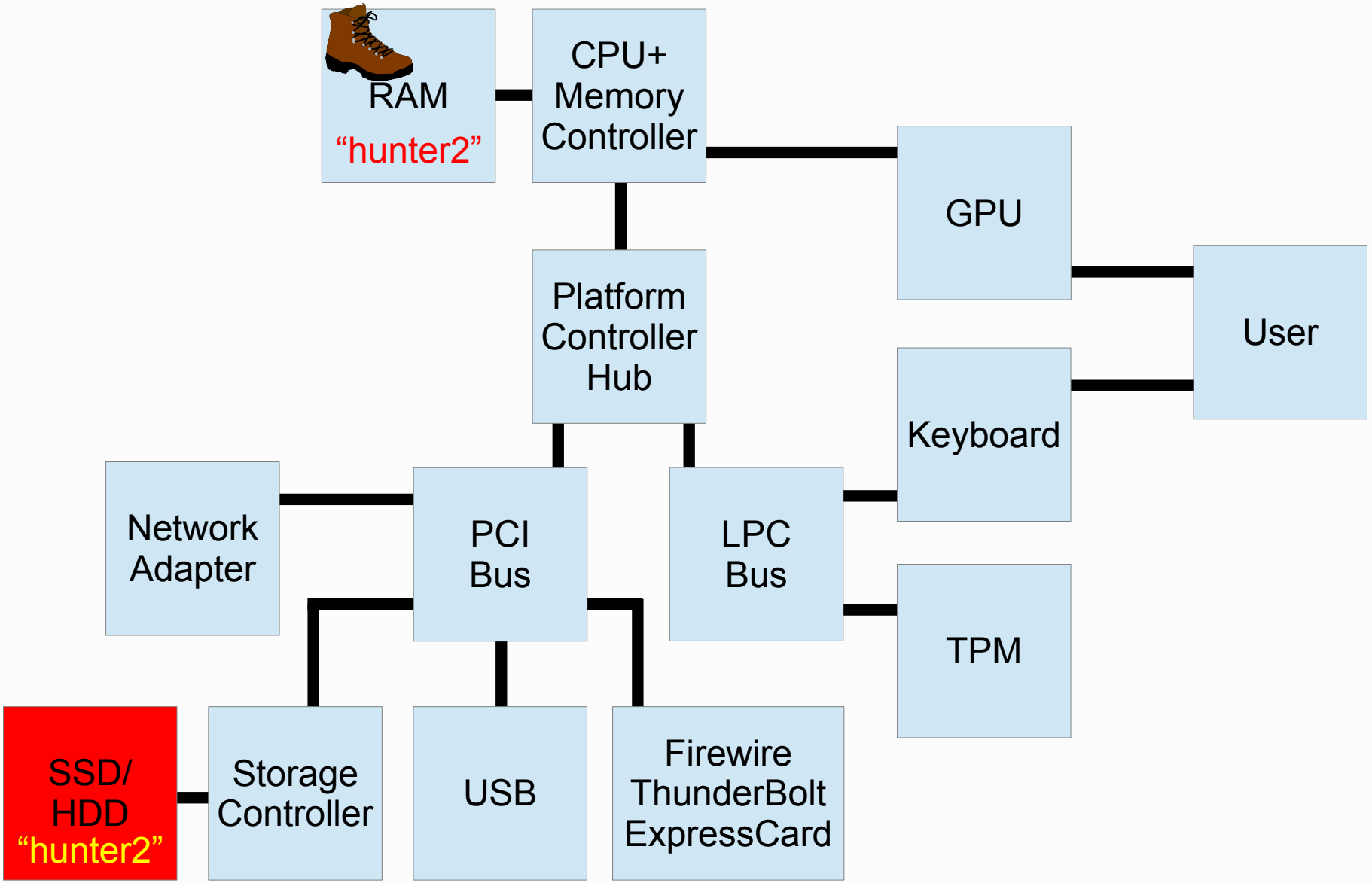
Three Attack Tiers

Non-Invasive

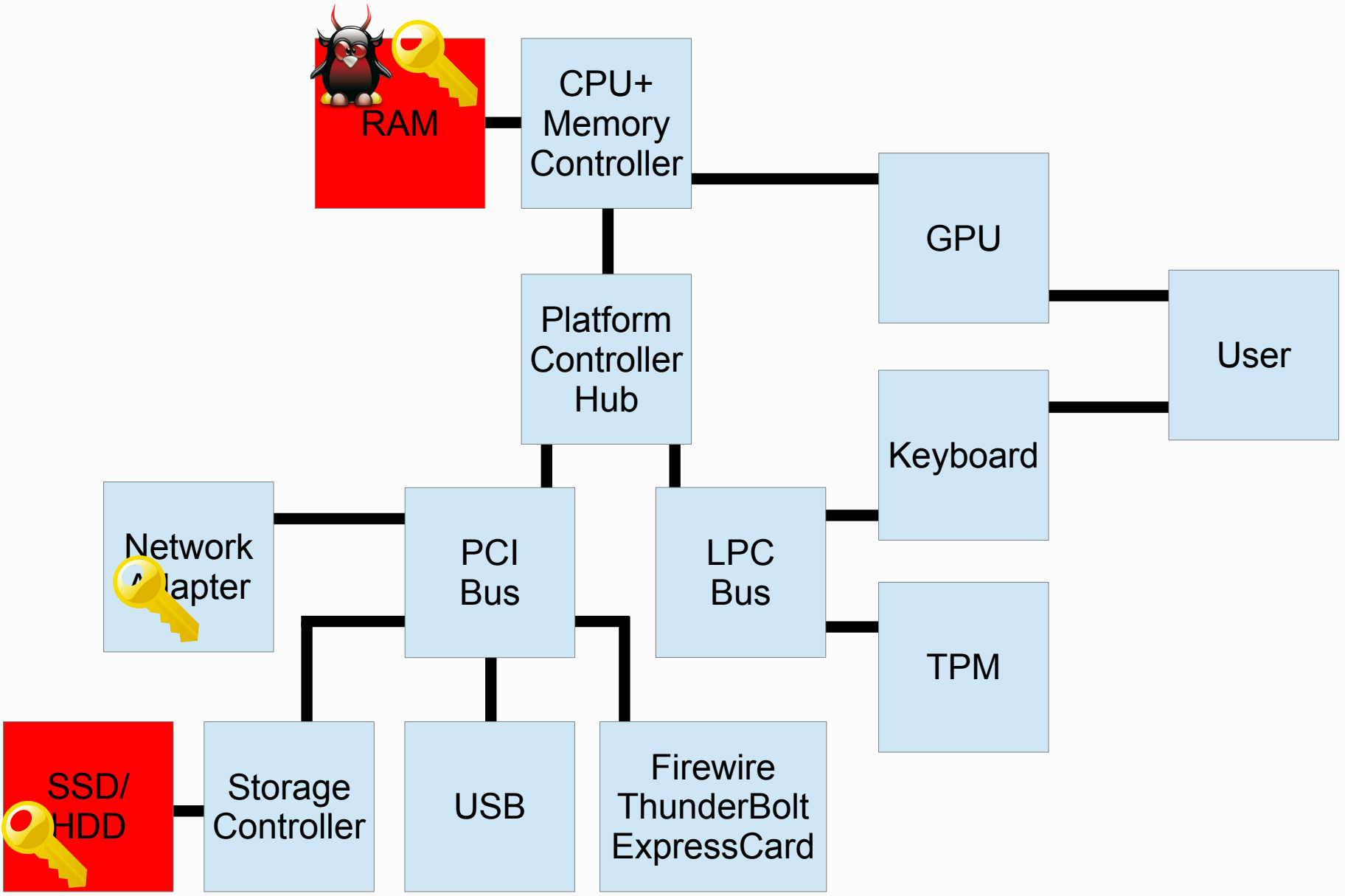
Screwdriver

Soldering Iron

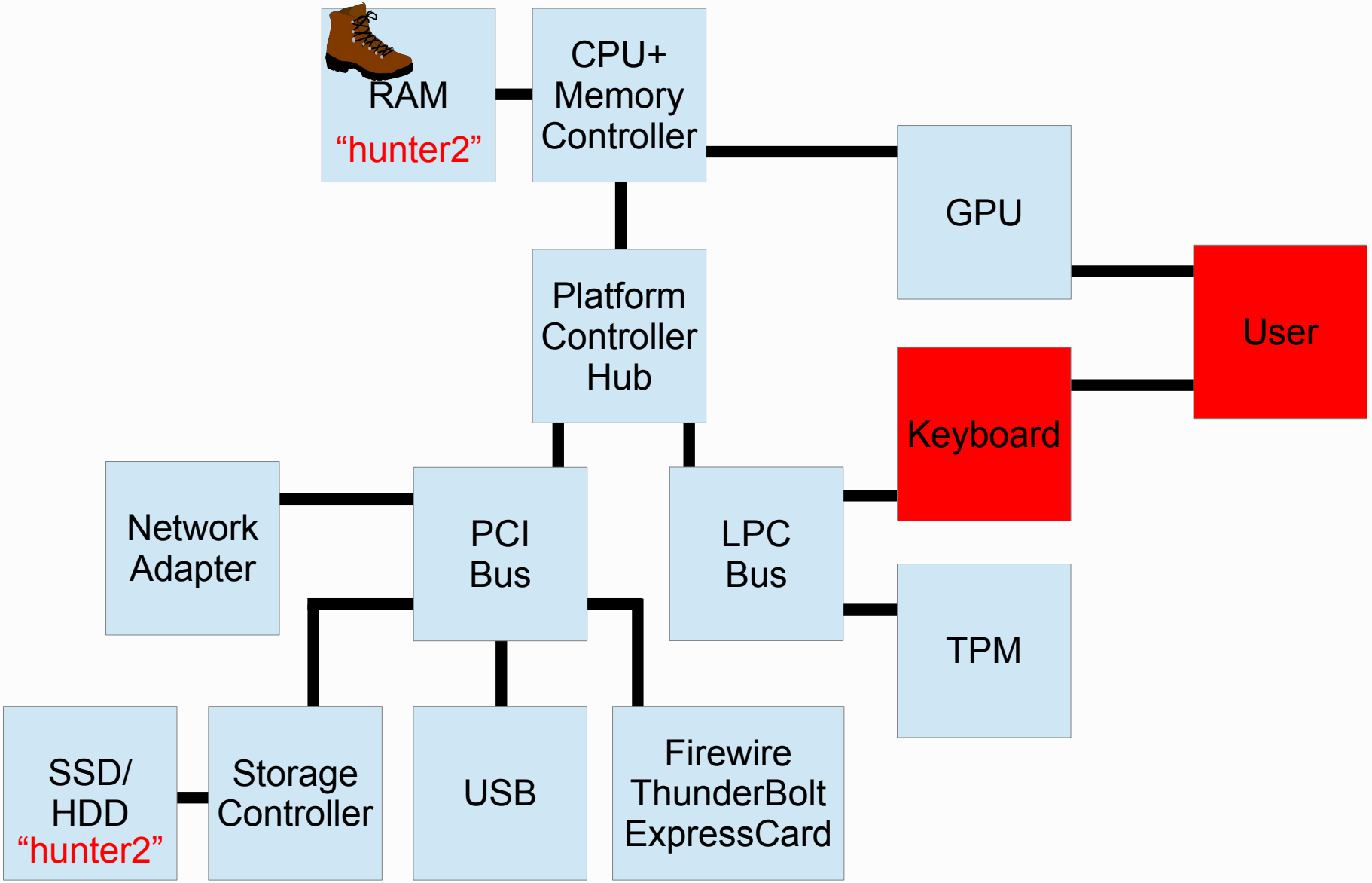
Compromised Bootloader



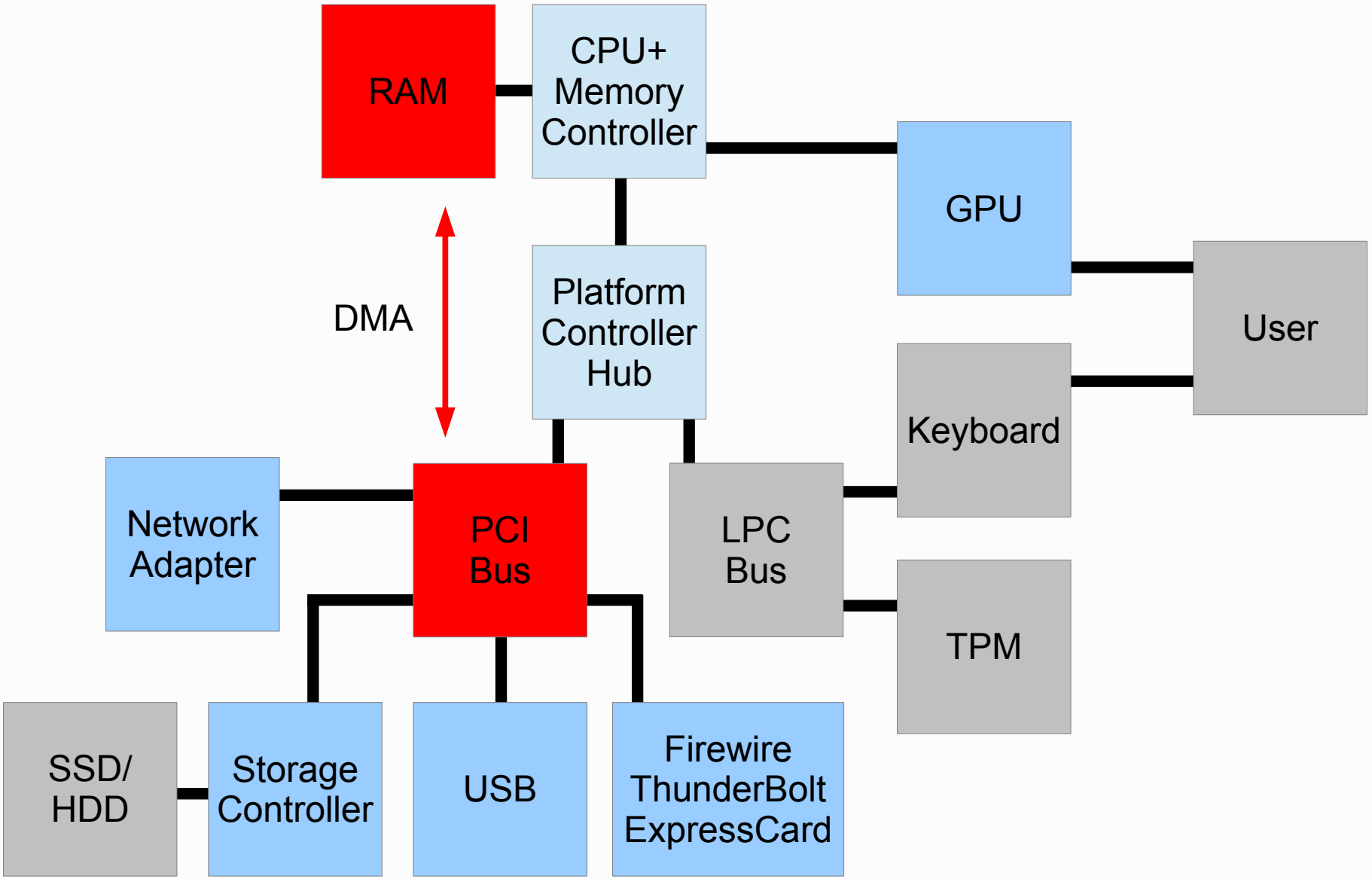
Compromised Operating System



Key Logger/Shoulder Surfing



Direct Memory Access



Cold Boot Attack

CPU+ Memory Controller

GPU

User

Platform Controller Hub

Keyboard

Network Adapter

PCI Bus

LPC Bus

TPM

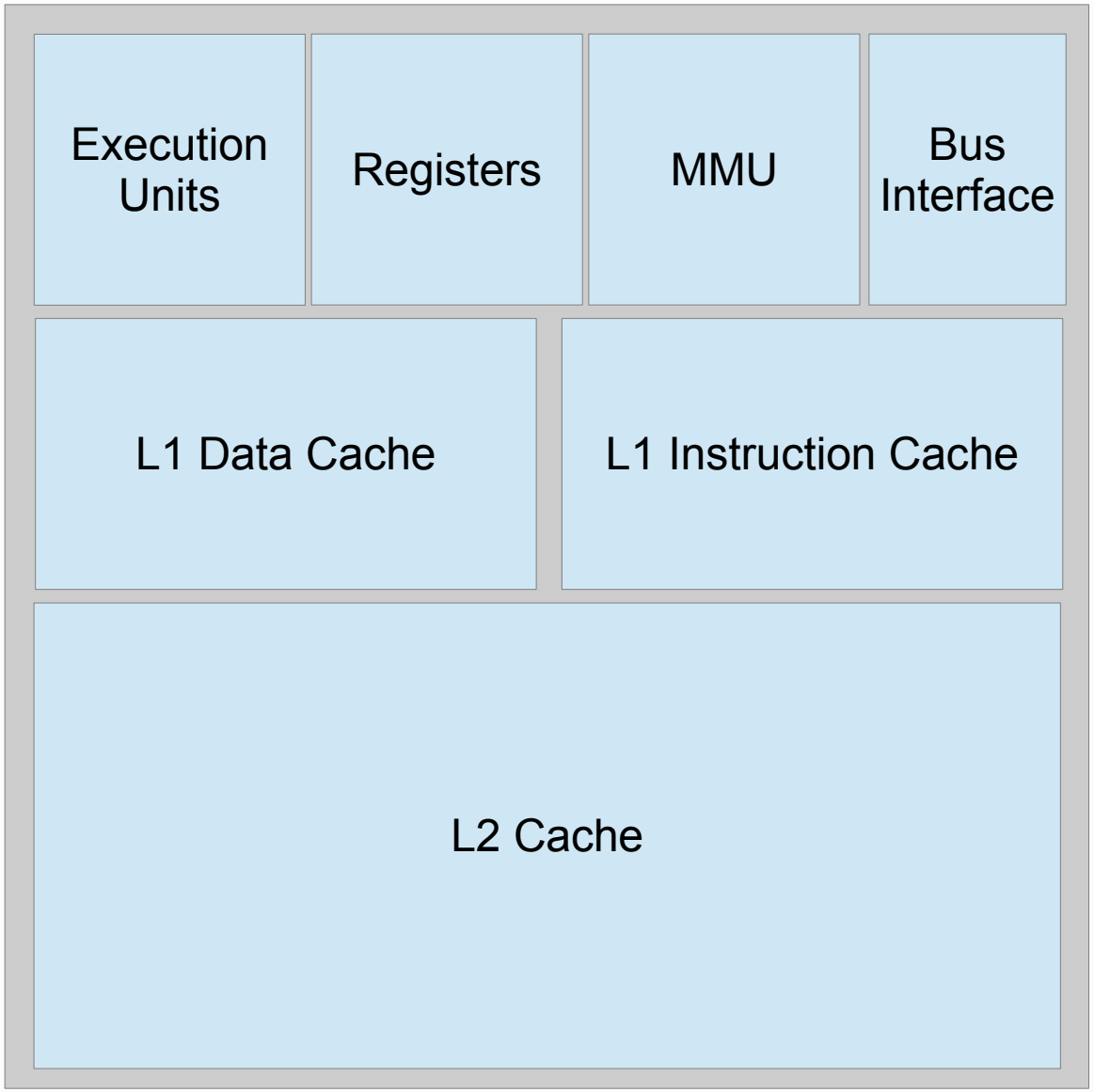
SSD/ HDD

Storage Controller

USB

Firewire ThunderBolt ExpressCard

What's in a CPU anyway?



x86-64 as “hardware crypto module”

- AES-NI (2010): hardware instructions for AES
- No known method of fixing cache lines
- Registers as key storage?
  - Machine State Registers (loop-amnesia)
  - SSE (AESSE)
  - DRx hardware debugging registers (TRESOR)
- Bonus: registers zeroed on ACPI S3 sleep

AES in x86-64 debug registers

- DRx not used in typical OS/software operation
- 6 registers
  - DR0-DR3 = 64-bit breakpoint addresses
  - DR6-DR7 = behavior/control/signal flags
  - 256-bits of key storage
- Use SSE registers as scratch for key schedule expansion
- First implemented by Tilo Müller as TRESOR for Linux in 2011

http://www1.informatik.uni-erlangen.de/tresorfiles/tresor.pdf

RAM still vulnerable

- Hostile DMA could alter OS to dump DRx
- Is there a way to restrict hostile DMA transfer?
  - IOMMU technology in the memory controller
  - Intel VT-d, AMD Vi
- Use IOMMU to protect “OS”
- TreVisor implements TRESOR on BitVisor
  - Transparent encrypting hypervisor for a single guest

http://www1.cs.fau.de/filepool/projects/trevisor/trevisor.pdf

Other sensitive data in RAM

- Active files are cached in RAM
- SSH/PGP keys, password manager DBs
- Encrypt everything you don't want to leak
- Self-encrypting drives are insufficient
- Can we encrypt RAM?

Encrypting RAM

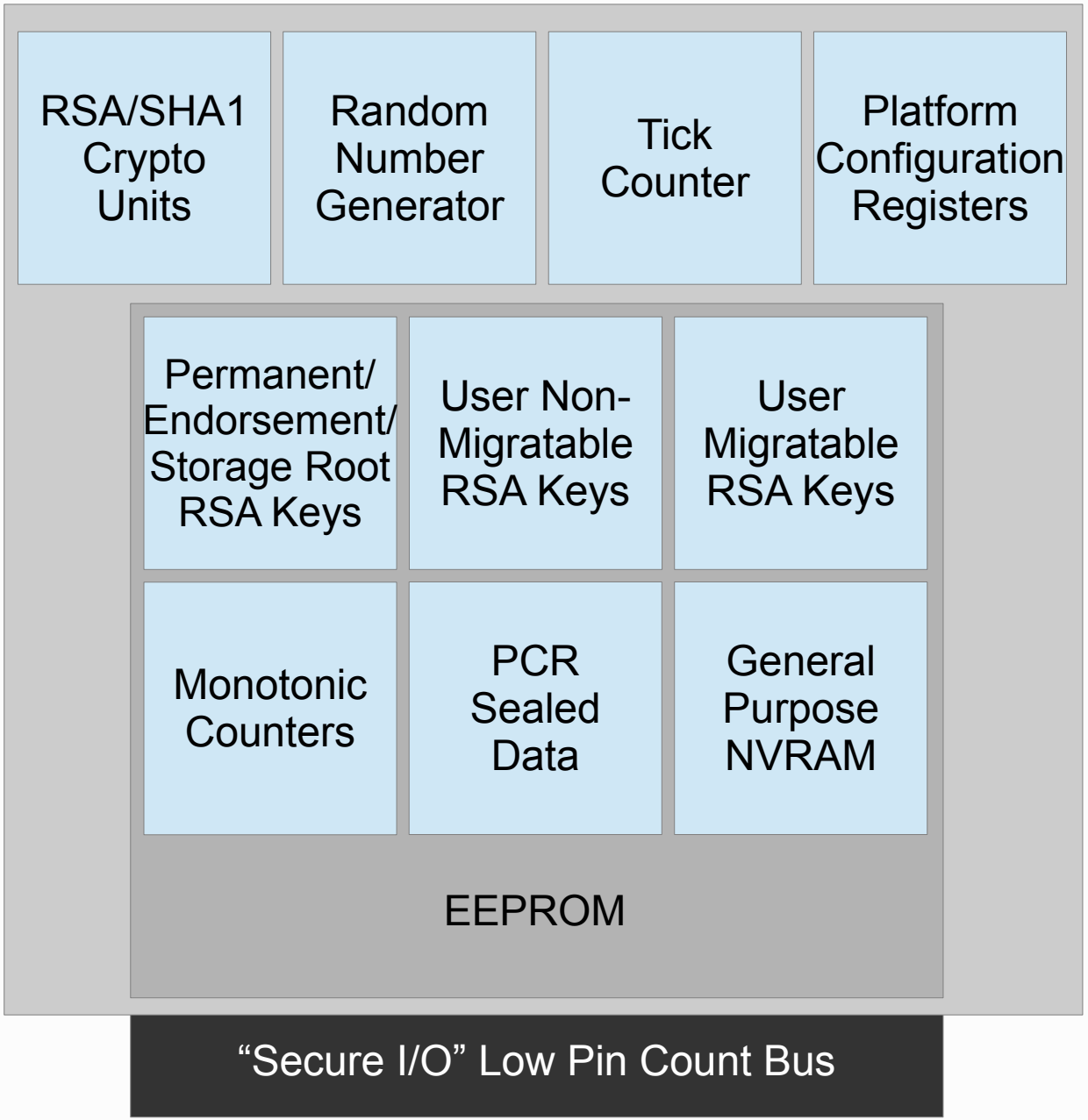
- CryptKeeper proof of concept by Peter Peterson
  - Divided RAM into small “clear” and larger “crypt”
  - Data moved using ordinary Linux paging facilities
  - 10x-50x slower in synthetic benchmarks
  - ~10% slower in “typical use” benchmark
  - “Crypt” key stored in “clear”
  - Author considered use of hardware crypto module

  Cryptkeeper: Improving Security With Encrypted RAM, 2010
  http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5655081

Verifying Computer Integrity

- User must be able to determine if their computer is pristine or tampered before authenticating themselves
- Trusted Platform Module can be leveraged for measured boot
  - Data can be sealed to values of “platform configuration registers”
  - “Extend” PCRs with stage measurements
    - Extend(x, payload) → PCR[x] = SHA1(PCR[x] + SHA1(payload))
  - SRTM: ROM extends with BIOS, BIOS extends with bootloader, bootloader extends with kernel/initrd, etc.
  - DRTM: Under IOMMU protection load/execute a payload
- Seal secrets (cryptographic or otherwise) to enable verifying the computer by the user

Trusted Platform Module



TPM Sealed Secrets

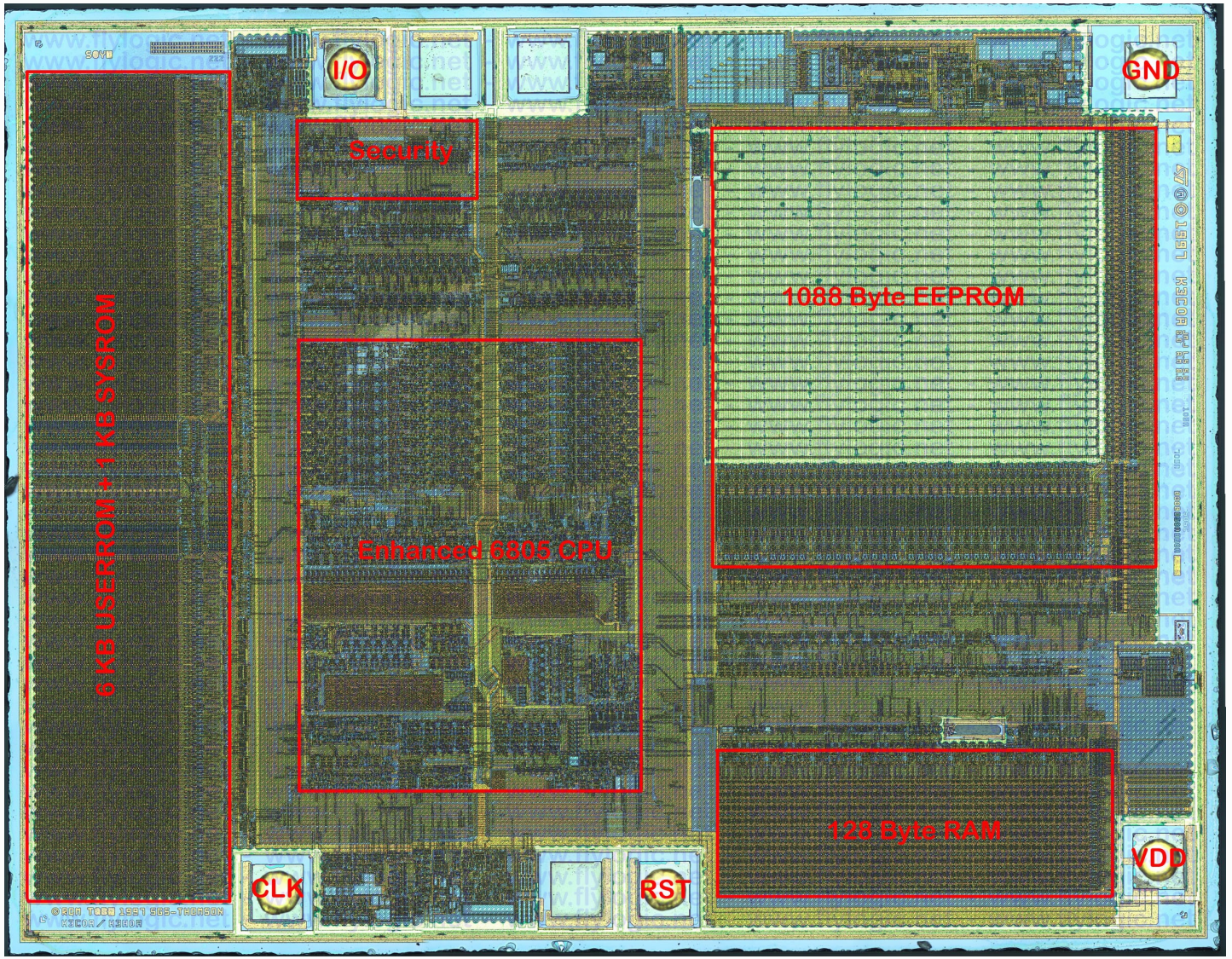
- Seeds for TOTP or HOTP tokens
- Recognizable and unique image/animation
- A part of the input to a key derivation function for the disk key
- By tying the disk key to the TPM, we can effectively dictate system boot policy


TPM Facilitated System Boot Policy

- Cloning the disk is of limited benefit/disk only accessible in origin computer
- Monotonic counters can be compared between on-disk and TPM NVRAM values
- Policy parameters defined by the user, e.g.:
  - Limit number of incomplete boots
  - Password entry timeout
  - Limit number of incorrect passwords entered
  - Limit time between last shutdown and subsequent boots
  - Entry of duress code
- Policy violations could be enforced brutally
  - Issue a TPM owner clear on violation

http://en.wikipedia.org/wiki/File:Burn-Bag.jpg

TPM vs Hardware Attacks



http://flylogic.net/chippics/st16601/st16601op_large.jpg

“Phalanx” Toolset

- 1) Patch to Xen hypervisor
  - Implements TRESOR variant
  - DR2/DR3 still available to VMs
  - DR0/DR1 used as “master” AES-128 key
  - Xen dom0 can “load key” to DR0/DR1
  - DR2/DR3 values encrypted with master key on guest context switch

“Phalanx” Toolset

- 2) Patches to Linux kernel
  - Modified TRESOR to work on AES-128 in DR2/DR3 only
  - Modified zRAM using TRESOR to encrypt pages after compression
- 3) Userspace utilities
  - Initrd script skeleton
  - Built on tboot and Intel TXT
- Get the source code: https://github.com/thyth/phalanx

Suggested Installation Basis

- Qubes OS (http://qubes-os.org/)
  - Pragmatic formulation of Xen, Linux, and custom tools to provide “security by isolation” model
  - Isolate information in separate domU guests in Xen
  - Desktop environment designed for seamless use of multiple VMs

Hardware Requirements

- AES-NI
- Hardware Virtualization Extensions
  - Intel VT-x, AMD-V
- IOMMU
  - Intel VT-d, AMD-Vi
- TPM
  - Static/Dynamic Root for Trust Measurement

Security Assumptions

- TPM:
  - No backdoor capable of dumping NVRAM
  - No backdoor capable of resetting monotonic counters
  - No effective “reset” attack on PCR state
  - Conducting hardware attack to tap TPM CPU and extract secrets should take no less than 12 hours

Security Assumptions

- CPU, Memory Controller, IOMMU
  - Correctly configured IOMMU is effective protection against hostile DMA transfer
  - AES-NI has no backdoor
  - HVM generates correct VMEXIT events
  - No Intel backdoor in TXT
- Xen
  - No (more) hypervisor compromise vulnerabilities
  - Correct implementation of PV hypercalls

Threat Model

- Realistic Threat Assessment:
  - No system is unbreakable, especially one that contains so many vulnerable parts
  - Safes are rated in number of minutes they can withstand an adversary
    - Aim for hours, not minutes
  - Assumptions can be wrong (verify mine!)

Expected Security

- Cold boot attack ineffective against FDE key, and encrypted user information in RAM
- Hardware based RAM acquisition ineffective
- Extracting TPM NVRAM will only re-enable “evil maid” attacks
- Tampering with the system sufficient to compromise security should be noticeable by the user (e.g. unseal fail, computer missing for hours)

Attack Methods

- Key loggers, cameras, and shoulder surfing
- TPM attacks
  - NVRAM extraction
  - LPC bus intercept/reset hardware
- RAM manipulation
  - Surreptitious RAM: hardware intercept/manipulation
  - Transient pulse injection

Legal Notes

- Not a lawyer, not legal advice
  - US 5th Amendment prevents compelling a password from a suspect in criminal cases
  - (In US) Automatic self destruct timer believed to be not illegal
  - TPM and strong cryptography illegal in some jurisdictions
  - Mandatory key disclosure in some countries

Future Work & Improvements

- Facilities for greater control over encrypted paging
  - Some data is more important than other data
  - Modify OpenSSL to aggressively swap out keys?
- Easily installable variant of the system
  - Based on Qubes OS
- Upstream the patches?
  - Unclear if they would be accepted

Conclusions

- The best security model in the world will go unused if it is unusable
- The security model must account for realistic use patterns
- Disk encryption is not enough: real protection comes from “full” system encryption
- System encryption is barely possible on commodity hardware
- But it's still better than the status quo

Thank you!

- Go get the code: https://github.com/thyth/phalanx
- Web: http://thyth.com/
- Email: ds@thyth.com
- PGP: ID 0xdfc02d75

  201a 7b59 a15b e5f0 bc37 08d3 bc7f 39b2 dfc0 2d75