Trusted Computing and Free Software
RMLL 2009 – Nantes

Frédéric Guihéry

AMOSSYS

July 9, 2009

Agenda

1. Let's define Trusted Computing
2. Misconceptions on TC
3. Current achievements
   - The Trusted Platform Module
   - Secure/Trusted Execution
4. Free softwares that leverage TPM
   - TPM utilities
   - Integrity measurement and verification
   - Cryptographic Filesystem
   - Network Authentication/Encryption
   - Secure/Trusted Execution
5. Usage analysis
6. Conclusion

Context

The author

- In the Free Software since 2002
- Member/sympathizer of different LUGs

Amossys

- Located in Rennes
- Expertise and consulting in architecture in information systems and security, IT Evaluation lab
- Contributor Member of the TCG

Let's define Trusted Computing

Let's define Trusted Computing

- Trusted property : we are sure of what is being executed at the moment of its launch
- Here, the term sure means we can measure and verify (either during or after the fact) its integrity
- This implies cryptographic operations
- Trusted environment or TCB : an environment where each component is trusted

Misconceptions on TC

Misconceptions on TC

- TCPA != Palladium != DRM
- The main papers against TC (see [1], [2] and [3]) refer mainly to TC-based-DRM and do not apply to all the TC aspects. Above all, they only deal with the first version of the TCG specifications.
- Neverthelesse, they were necessary at this time in order to counter the potential treacherous goals of some companies

Some excerpts

- "Proprietary programs will use this device to control which other programs you can run,..." [1]
- "..., the TCG specification will transfer the ultimate control of your PC from you to whoever wrote the software it happens to be running." [2]
- "It could prevent the use of "free" operating systems because the OS kernel would have to be signed by a entity which is a descendant of the trusted root." [3]

[1] Can you trust your computer ?, R. M. Stallman
[2] Trusted Computing FAQ, R. Anderson
[3] The TCPA; What's wrong; What's right and what to do about, W. A. Arbaugh

Current achievements

Current achievements

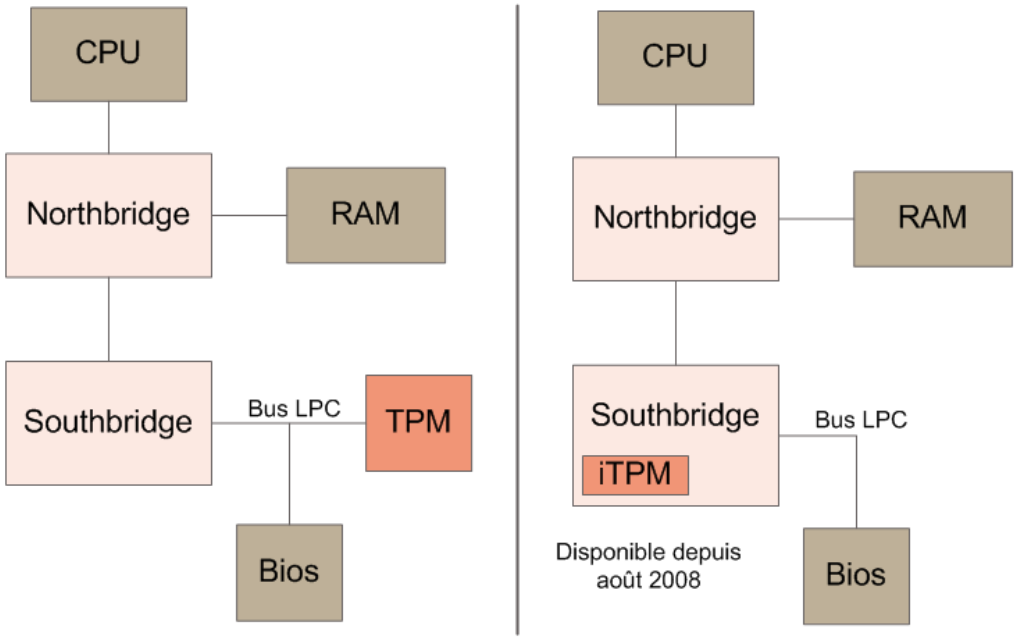
Goal of the TCG : creating open security standards.

- Trusted Platform Module (TPM) :
  - specifications as an ISO standard (TCG – 2000/2006)
  - implementations (chip manufacturers)
- Trusted Network Connect (TNC) :
  - specifications (TCG – 2008/2009)
  - few implementations (network manufacturers)
- Secure/Trusted Storage :
  - specifications (TCG – 2007/2009)
  - few products (disk manufacturers)
- Secure/Trusted Execution
  - specifications and implementations (made independently by semiconductor chip makers – 2007/2008)

The Trusted Platform Module

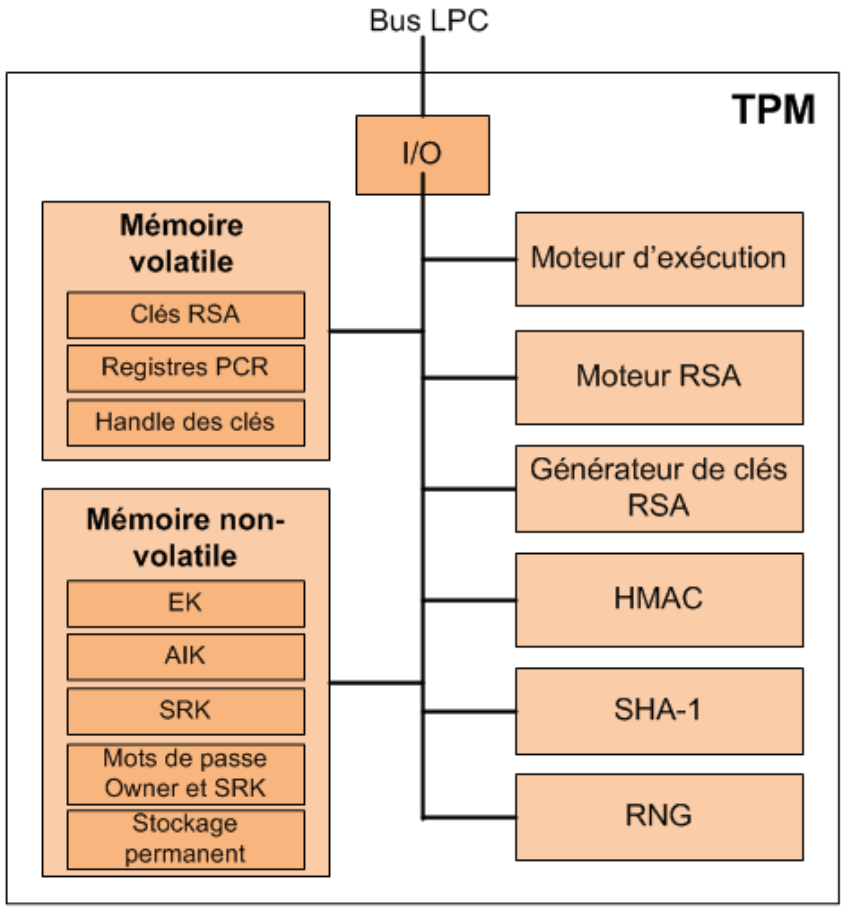
The Trusted Platform Module

- Slave crypto-processor connected on the LPC bus of a motherboard
- The TPM has no control on the system execution, nor can he monitor it
- Only manipulates crypto materials (keys, hashes, encrypted data) and has no comprehension on the origin of the data or its semantic
- The TPM can be deactivated and administrated by the platform owner
- Main manufacturers : Infineon, Atmel, Broadcom, STM, Intel, etc.
- Recently incorporated directly in the southbridge (chipset Intel ICH10)



TPM functionalities

- Random generator
- Key management
- RSA encryption/signature
- SHA-1 hash and HMAC functions
- PCR register with SHA-1 values
- that can only be extended
- Signature of the PCR values
- Cryptographic operations can be bound to a specific TPM and/or state of the PCR
- Etc.



Advantages and Drawbacks

Advantages

- Cryptographic operations done inside a hardware device
- The private RSA key can't leave the TPM in clear
- Base for robust security applications

Drawbacks

- Beyond the public specifications, the internal implementation is done as a black box
- No symmetric encryption
- Cryptographic operations are pretty slow
- The cryptographic manipulation of a huge amount of data has to be done outside the TPM (thus, the session key is available in the system memory)

Typical TPM usage : SRTM

Context :

- How to trust the current security root (i.e. the kernel) on a PC ?
- How to detect if a PC has been compromised (remotly, locally or even physically) with a rootkit/keylogger and so on, since the first installation ?

A first solution :

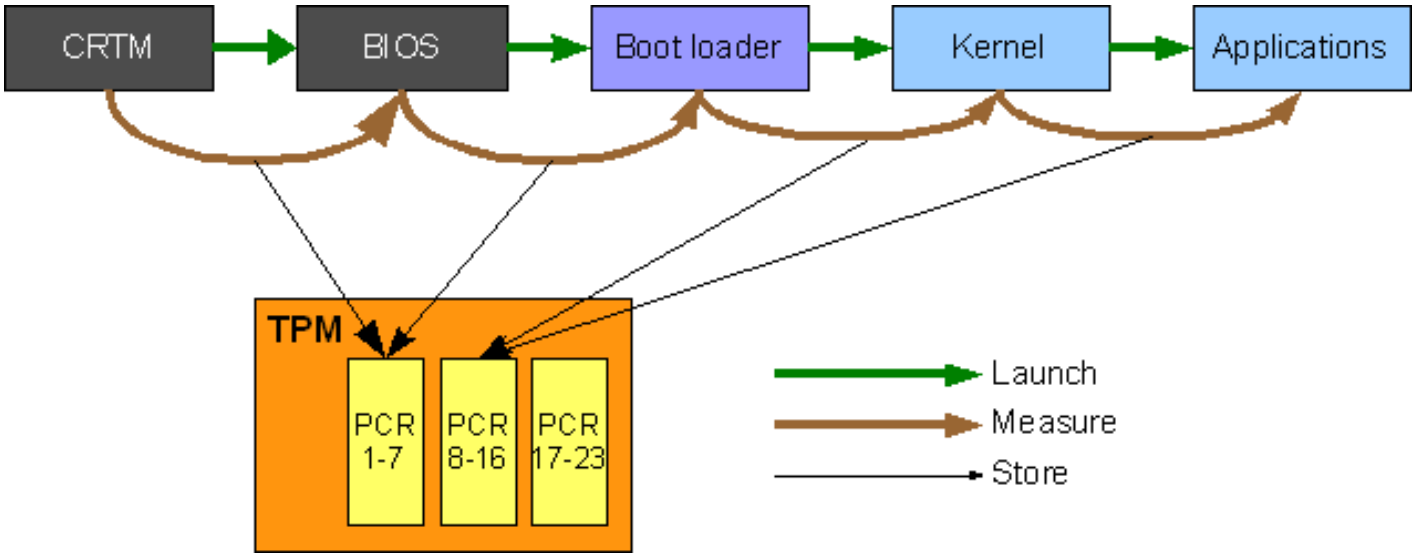
- Booting with a live-CD and measuring each software component. And then, comparing the measurements with the original ones.

Another solution would be to realize the same thing, but for each boot of the PC

- This is what a SRTM (Static Root of Trust Measurement) is trying to do

Typical TPM usage : SRTM

- The goal of a SRTM is to measure the integrity of each software elements started from the early boot
  - This process is initiated by the CRTM/BIOS which is the core root of trust
  - Integrity measurements are stored in PCR registers (*extend* function)
  - Scheme security = Robustness of SHA-1 & Unbreaking of trust chain
  - Cryptographic operations done inside a hardware device (the TPM)



Typical TPM usage : SRTM

But, this is still not sufficient...

- How to handle verification of measurements ?
- How to avoid binding the measurement to the underlying hardware (BIOS, microcode, etc.) ?

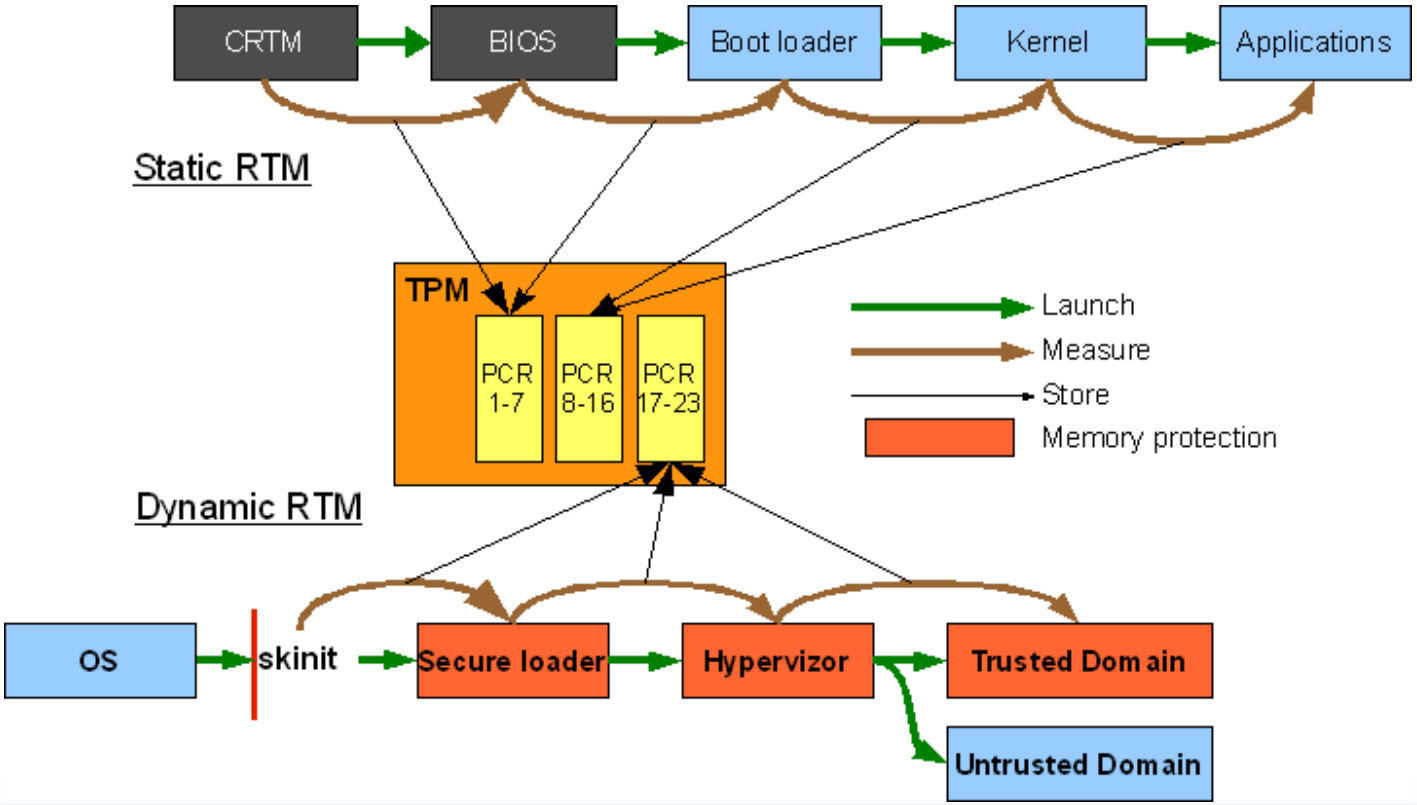
Secure/Trusted Execution

Secure/Trusted Execution

- Context
  - Besides the above limits of SRTM, the user could want to run sensitive code inside an environment known to be secure (i.e. the hability of running a trusted domain in parallel of an untrusted domain)
- Key points
  - Dynamic launch of a trusted environment
  - Trusted execution environment
  - Memory protection of this trusted environment
- Required technology
  - TPM chip
  - Hardware Virtualization support (Intel VMX or AMD SVM)
  - "Trusted Launch"-supported processor
    - Intel TXT / SMX : Trusted eXecution Technology / Safer Mode Extensions
    - AMD SVM / Presidio : Secure Virtualization Mode (with skinit instruction)
  - IOMMU-supported chipset

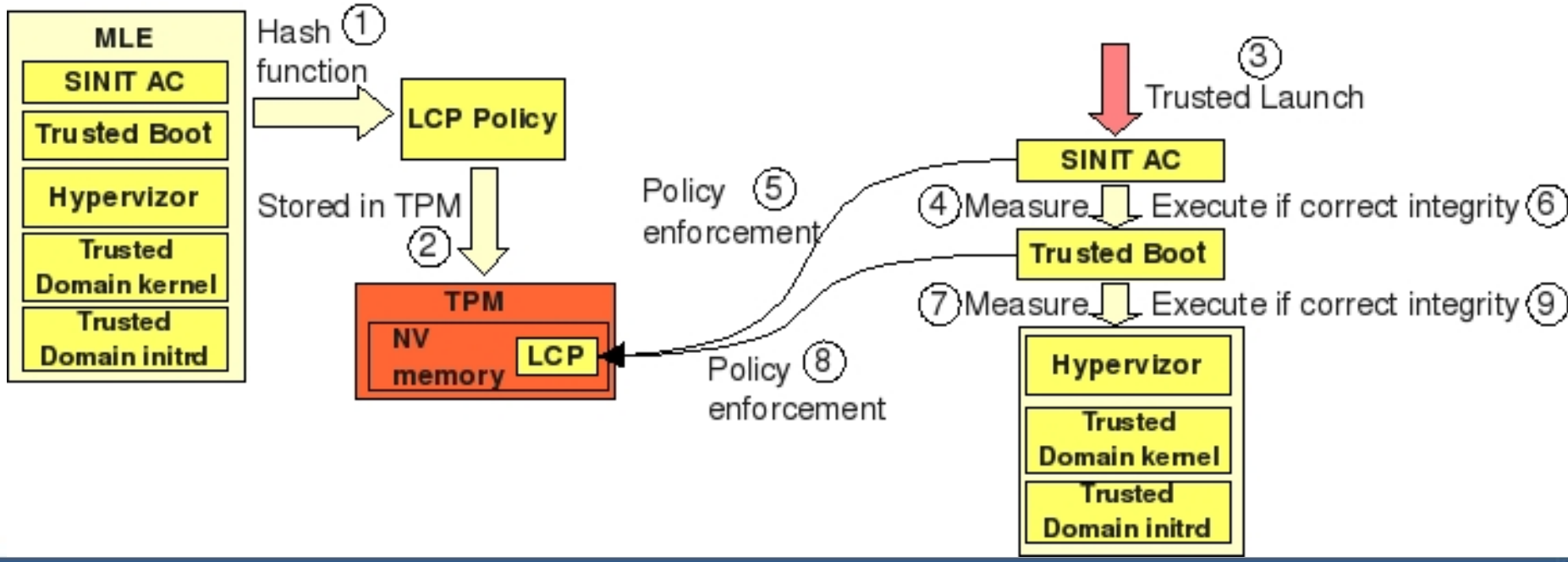
Secure/Trusted Execution

- Underlying mechanism
  - DRTM : Dynamic Root of Trust Measurement
  - DMA Protection with IOMMU



Secure/Trusted Execution

- Underlying mechanism
  - Security bonus on Intel platform (Intel TXT/SMX) : Launch Control Policy
    - Integrity of a known state saved in a policy
    - Next boot or next DRTM: integrity measurement and policy enforcement
  - Works with Linux and Xen (see picture)

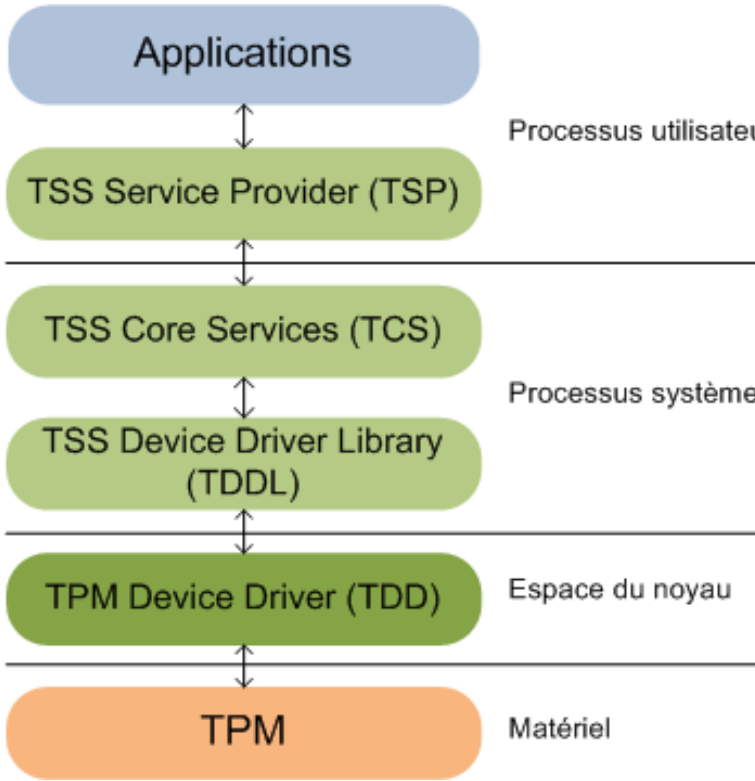


Free softwares that leverage TPM

TPM utilities

TPM utilities : TrouSerS

- TrouSerS : TSS under Linux developed by IBM[a]
- Licence : Common Public Licence
- Available in Debian, Gentoo, Fedora, etc.
- Provides :
  - Communication with the TPM
  - Synchronization of each application requests
  - Key management (public key)
  - User/Owner authentication
- Leverage TPM communication protection
  - Authorization Protocol : integrity protection + mutual authentication of TPM/User
  - Transport Sessions (TPM 1.2) : confidentiality protection



[a] http://trousers.sourceforge.net

TPM utilities : TPM tools / TPM/J

- TPM tools[4]
  - Few tools that leverage TPM functionalities
  - Made by IBM
  - Licence : Common Public Licence

- TPM/J[5]
  - Java TPM API made by MIT PhDs
  - Licence : BSD (some parts in Public Domain)

[4] http://trousers.sourceforge.net/man.html
[5] http://projects.csail.mit.edu/tc/tpmj/

Integrity measurement and verification

IMA

- IMA (Integrity Measurement Architecture) is an integrity service provider
- IMA runs in Linux kernel and can
  - Measure integrity of loaded binaries (executable, drivers, shared libs)
  - detect integrity alteration in binaries
  - detect integrity violation
- IMA included in Linux since kernel 2.6.30
- Developed by IBM

IMA

- Mechanism
  - The kernel measures each binary executed at the moment of its launch
  - The kernel maintains a measurements database and in the same time extends the measurements in the TPM
- IMA is not an integrity verifier nor an integrity policy enforcer
  - This step can be done by the EVM (Extended Verification Module)
  - Or by a third party, with the help of the TPM signature (Remote Attestation)

Cryptographic Filesystem

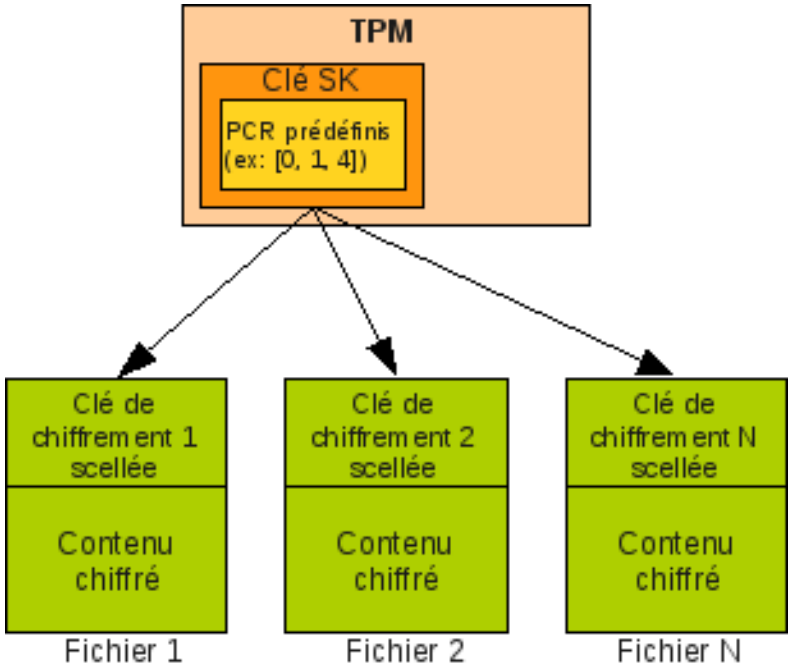
eCryptfs

- Cryptographic filesystem[6] (not a block device encryption like dm-crypt or Bitlocker)
  - Protect confidentiality against hard disk stealing
  - Protect against unauthorized access (other platform users of booting with live-cd)
- TPM interests
  - Protection of encryption keys in hard
  - Access to filesystem (unsealing of session keys) depends on the computer integrity

[6] https://launchpad.net/ecryptfs

eCryptfs

- Mechanism
  - One symmetric session key by file
  - Each session key is sealed by the TPM and stored in file header
- Status
  - Mainly written by IBM and Canonical developers
  - TPM support not mature at this time



Network Authentication/Encryption

Network Authentication/Encryption

- OpenTC PKI / PrivacyCA[7]
  - Provide a Privacy CA for use in Public Key Infrastructure
  - Made by IAIK from the Tugraz Institute
  - Licence : GPL
- OpenSSL TPM Engine[8]
  - Encryption/Signature of SSL flow with TPM keys
  - Made by IBM
  - Licence : GPL
- EAP-TPM protocol implementation[9]
  - FreeRADIUS server, wpa_supplicant clien, OpenSSL TPM Engine
  - Made by Carolin Latze from the University of Fribourg

[7] http://trustedjava.sourceforge.net

[8] http://sourceforge.net/project/showfiles.php?group_id=126012&package_id=165637

[9] http://diuf.unifr.ch/people/latzec/prototyping/first/

Network Authentication/Encryption

- TPM interests in network flow protection
  - Protection of encryption keys in hard
  - More reliable for mutual authentication
  - Combined with PC measurement -> allow the autorisation of the connection to a local network if the integrity is correct
    - Avoid the compromission of other PC clients on the network

Secure/Trusted Execution

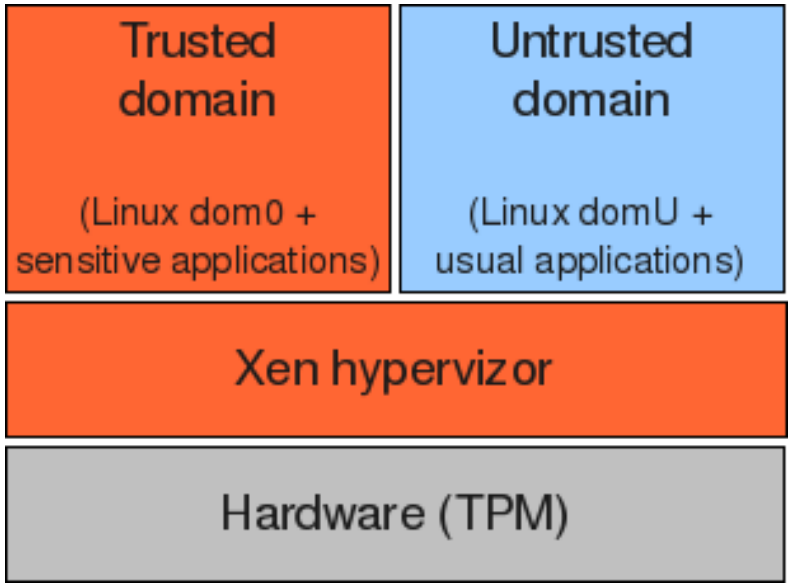
Secure/Trusted Execution

- Goal : running a trusted domain in parallel of an untrusted domain
- How ? : implementation of DRTM with domain memory protection
- Software components
  - Linux[10]
    - Supports Intel TXT with a patch proposed in 2.6.30
    - Licence : GPL
  - or Xen[11]
    - Virtualization project from Cambridge University
    - Licence : GPL
  - Trusted Boot[12]
    - Secure boot loader from Intel
    - Licence : GPL

[10] http://www.kernel.org
[11] http://www.kernel.org
[12] http://sourceforge.net/projects/tboot/

Secure/Trusted Execution

- Mechanism
  - Trusted Boot acts as a pre-kernel
  - Realize a verified launch of the MLE (Hypervizor and dom0) with Intel TXT
- Typical usage
  - Security sensitive operations done inside Trusted domain
    - Network encryption
    - Firewall
    - Antivirus/IDS that protect untrusted kernel/apps
  - Usual operations done inside untrusted domain



Usage analysis

Analysis : Remote attestation

Remote attestation (in a TC-based-DRM context)

- Feasible theoretically but not in practice on usual environments
  - PKI doesn't scale worldwide
  - Measurements database doesn't scale easily
  - DRM usage is decreasing (no interesting market and problem of perdurance)
- Remote attestation is only applicable in specific contexts
  - Inside a company infrastructure
  - For remote hardware like set-top-boxes
  - When the content provider is also the software/hardware manufacturer

Analysis : Disk encryption

Disk encryption

- Robustness of the keys protection
- Problem in case of legitimate hardware modification which implies integrity alteration => the hard disk becomes undecipherable
- Need for key management and recovery

Analysis : Integrity verification

How to handle verification ?

- With a robust comparison (Intel TXT/LCP) from a previous known good state (1)
- Implicitly with an unseal that depends on a previous known good state (2)
- With a third party (3)

Then, how to propagate the trust verification from the system to the user ?

- In case 2, the trust state is also implicit
- In case 3, the trust state has to be retrieved on the third party system
- How about the case 1 ?
  - « If it's running, it's safe» ?

Analysis : Trusted Execution

In a non-virtualized context

- Help ensure the integrity state of the system before performing sensitive operations

In a virtualized context

- Can help protect against apps and kernel malware that try to compromise the untrusted domain
- Works better with small dom0 / hypervizor (less exposure to vulnerabilities)

Other potential usage

- Regular integrity verification at runtime

Conclusion

This presentation tried to expose the following points:

- There is a current TC development in the Free Software field, mainly carried out by academic labs and IBM / Intel folks
- There is a gain in term of security for legitimate TC protection
- Illegitimate or unethical usages are theoretically feasible but practically difficult to deploy (except in some closed contexts)
- A technology should not be directly considered as harmful without considering the realistic usages that can be built on it

Questions ?