



État des lieux de l'informatique de confiance

SAR&SSI, La Rochelle – 20/05/2011

**Frédéric Guihéry
AMOSSYS**



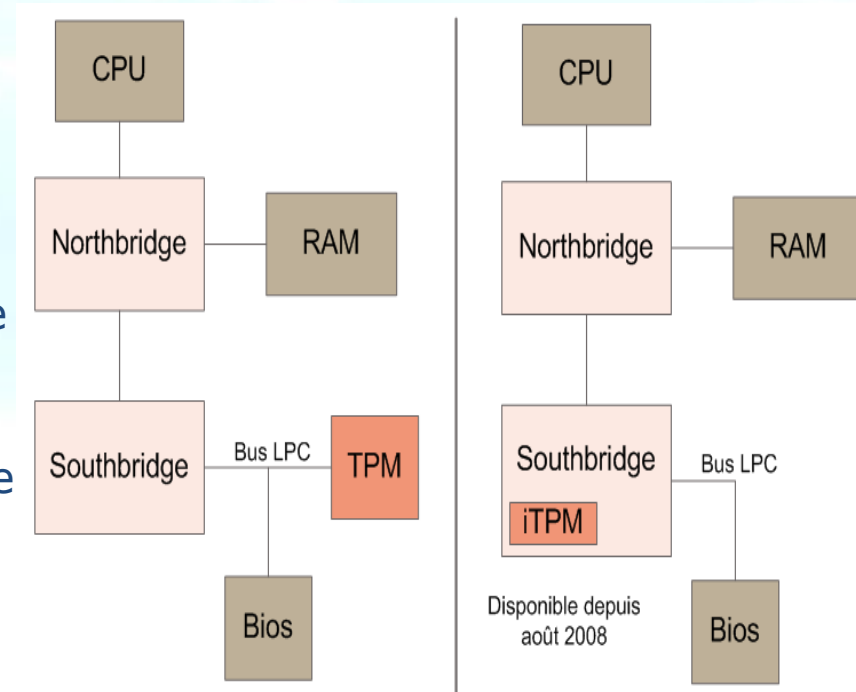
- AMOSSYS
 - Localisé à Rennes
 - Expertise et conseil en sécurité des systèmes d'information
 - Laboratoire d'évaluation (CESTI / CSPN)
 - Membre contributeur du *Trusted Computing Group*



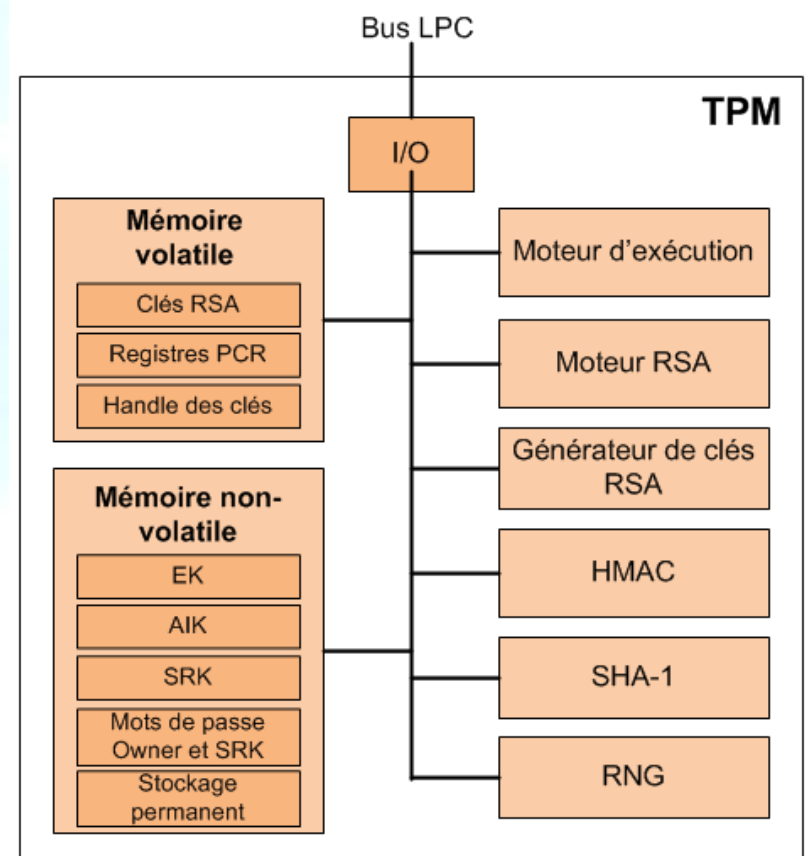
Plan du tutorial

- État des lieux de la technologie TPM
 - Architecture et fonctionnalités
 - Focus sur les produits IMA et EVM
 - Bilan
- État des lieux de la technologie Intel TXT
 - Principes
 - Focus sur le produit Trusted Boot
 - Attaques et contremesures
 - Bilan

- **Crypto-processeur esclave** connecté sur le bus LPC de la carte mère
- Le TPM n'a **aucun contrôle sur l'exécution du système**
- Il ne manipule que du **matériel cryptographique** (clés, hachés, données (dé)chiffrées) et n'a aucune compréhension de l'origine des données ni de leur sémantique
- Le TPM est activable et administrable par le propriétaire de la plate-forme
- **Principaux fondateurs** : Infineon, Atmel, Broadcom, STM, Nuvoton, Intel, etc.



- Cryptographie RSA
- SHA-1
- HMAC
- 24 registres PCR de 160 bits
- Générateur d'aléa
- Mémoire volatile
- Mémoire non-volatile



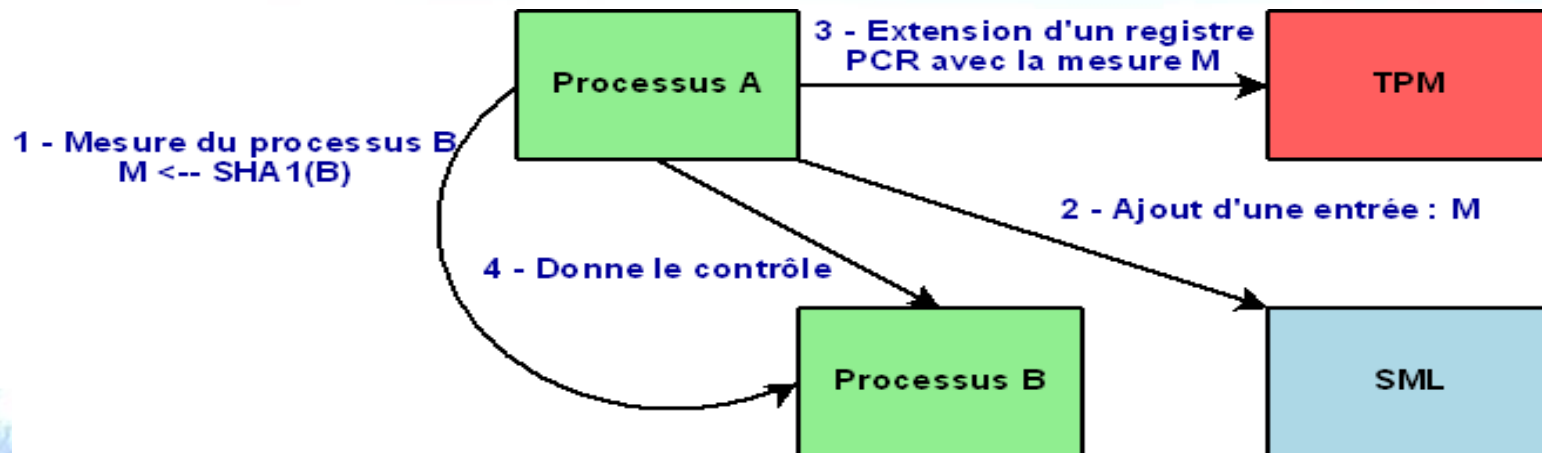
Technologie TPM

Le principe de mesure d'intégrité

- Extension des registres PCR

$$PCR[t+1] \leftarrow \text{SHA-1}(PCR[t]||M)$$

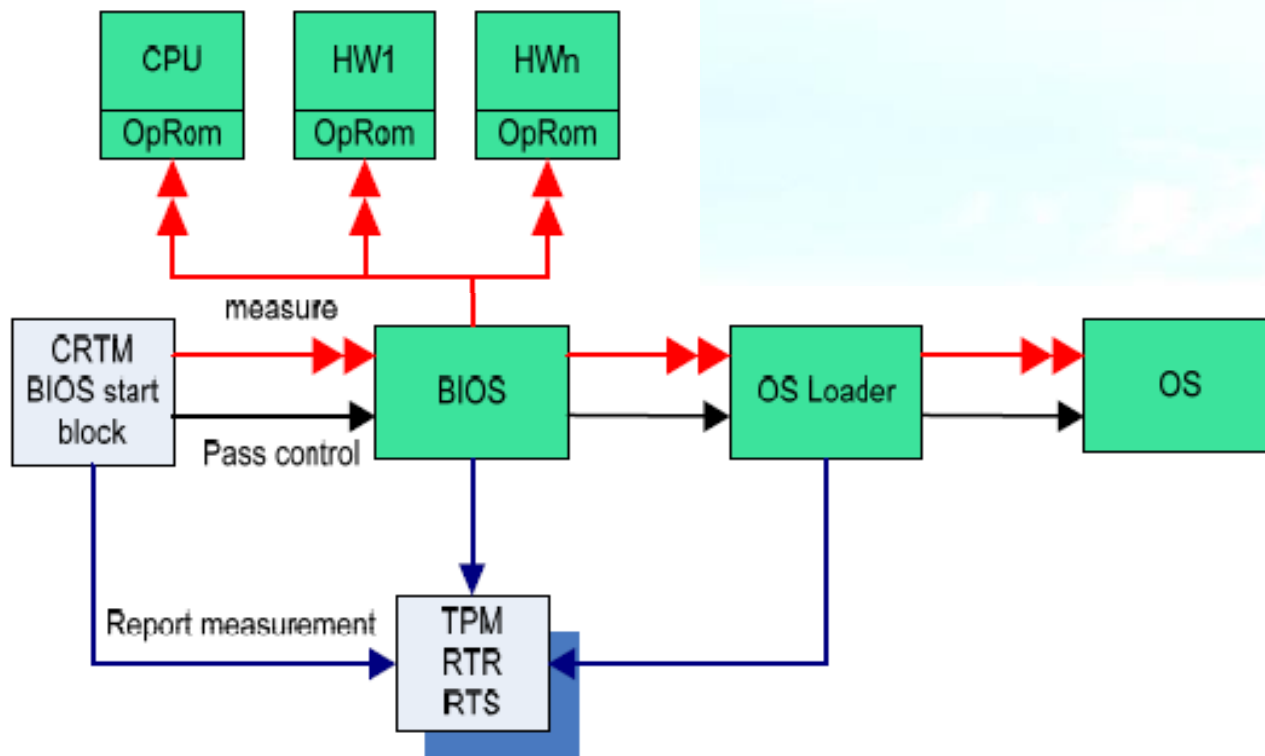
$$M = \text{SHA-1}(\text{DATA})$$



Technologie TPM

Le principe de chaine de confiance

- Transitive Trust : chaque élément de la chaine de démarrage mesure l'intégrité de l'élément suivant avant de l'exécuter





Technologie TPM

Le principe du scellement

- Scellement := chiffrement conditionnel
- Principe :
 - Si $\text{Etat}_{\text{PCR scellement}} == \text{Etat}_{\text{PCR descellement}}$
 - alors libération de la donnée scellée
- Conditionnement par rapport à 1 ou plusieurs PCR



Technologie TPM

Les usages

- Mesure d'intégrité
 - Base pour du chiffrement conditionnel ou de l'attestation
- Stockage sécurisé (chiffrement conditionnel)
 - Applicable à plusieurs niveaux : partition système, système de fichiers, fichiers
- Trousseau de clé sécurisé
 - Base pour des produits reposant sur l'identité de la plateforme/de l'utilisateur
- Vérification d'intégrité au démarrage
 - D'un OS
 - D'une application



Focus sur IMA et EVM



Technologie TPM

IMA : *Integrity Measurement Architecture*

- Module Linux développé par IBM
- Intégré dans Linux depuis la version 2.6.30
- Objectif : permet d'identifier des pertes d'intégrité
- Fonctionnement : mesure l'intégrité des binaires à chaque chargement
 - Pilotes
 - Exécutables
 - Librairies dynamiques
- Résultat dans un fichier de journalisation
- Mais module passif :
 - Ne réalise pas d'action (contrôle d'accès, vérification d'intégrité, etc.)



Technologie TPM

IMA-appraisal

- Patch noyau pour IMA développé par IBM
- Ajout d'un attribut étendu sur les fichiers : `security.ima`
- Stockage de la mesure d'intégrité SHA-1 du fichier dans cet attribut
- Vérification du haché à chaque accès au fichier

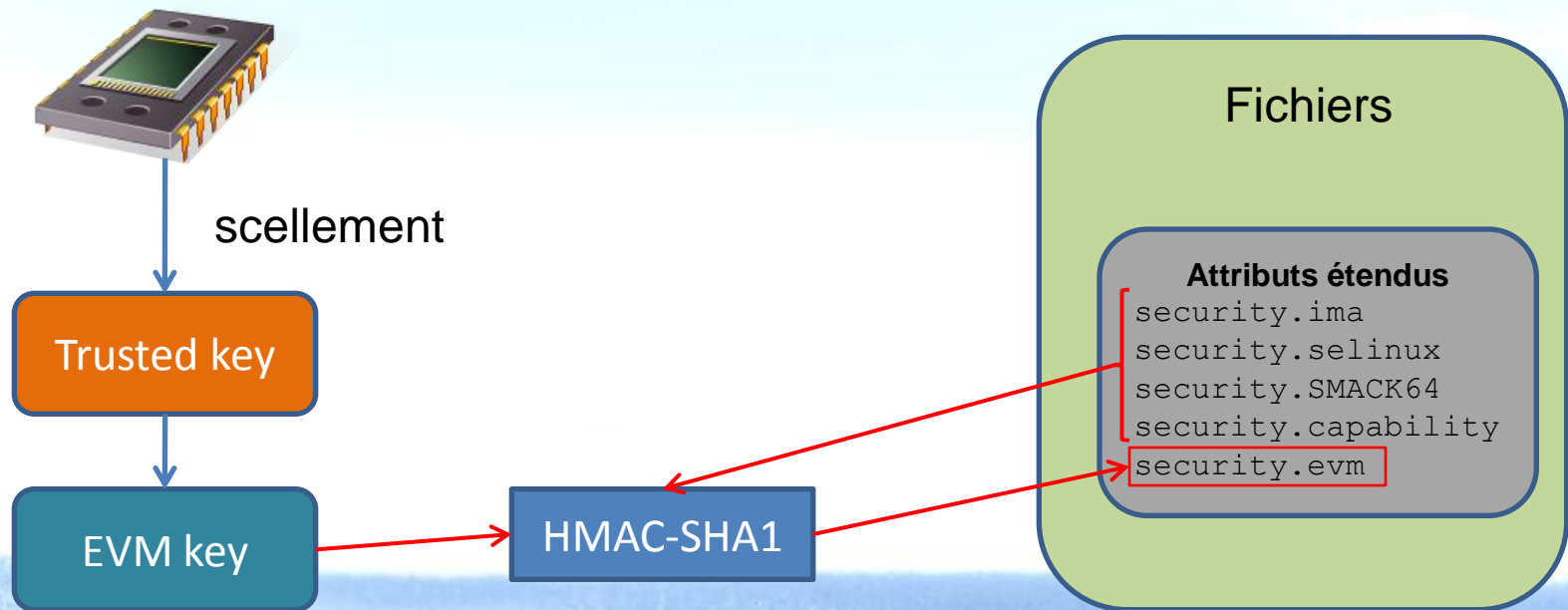


Technologie TPM

Trusted and Encrypted Keys

- Module pour le gestionnaire de clés du noyau Linux (2.6.38)
 - *Trusted Key* :
 - ✓ Clé AES scellée par le TPM
 - ✓ Interface flexible :
 - Possibilité de resceller un objet
 - Possibilité de définir la clé parente (par défaut : clé SRK avec mode de passe connu)
 - *Encrypted Key (EVM Key)* :
 - ✓ Clé AES protégée par une autre clé (une *trusted key* par exemple)
- Première brique pour EVM

- Patch du noyau Linux (bientôt intégré)
- Objectif : vérifier l'intégrité des fichiers au chargement
- S'appuie sur les *Trusted* et *Encrypted Keys*
- Ajout d'un attribut étendu des fichiers : `security.ima`
- Stockage du HMAC de certains attributs étendus :





Technologie TPM

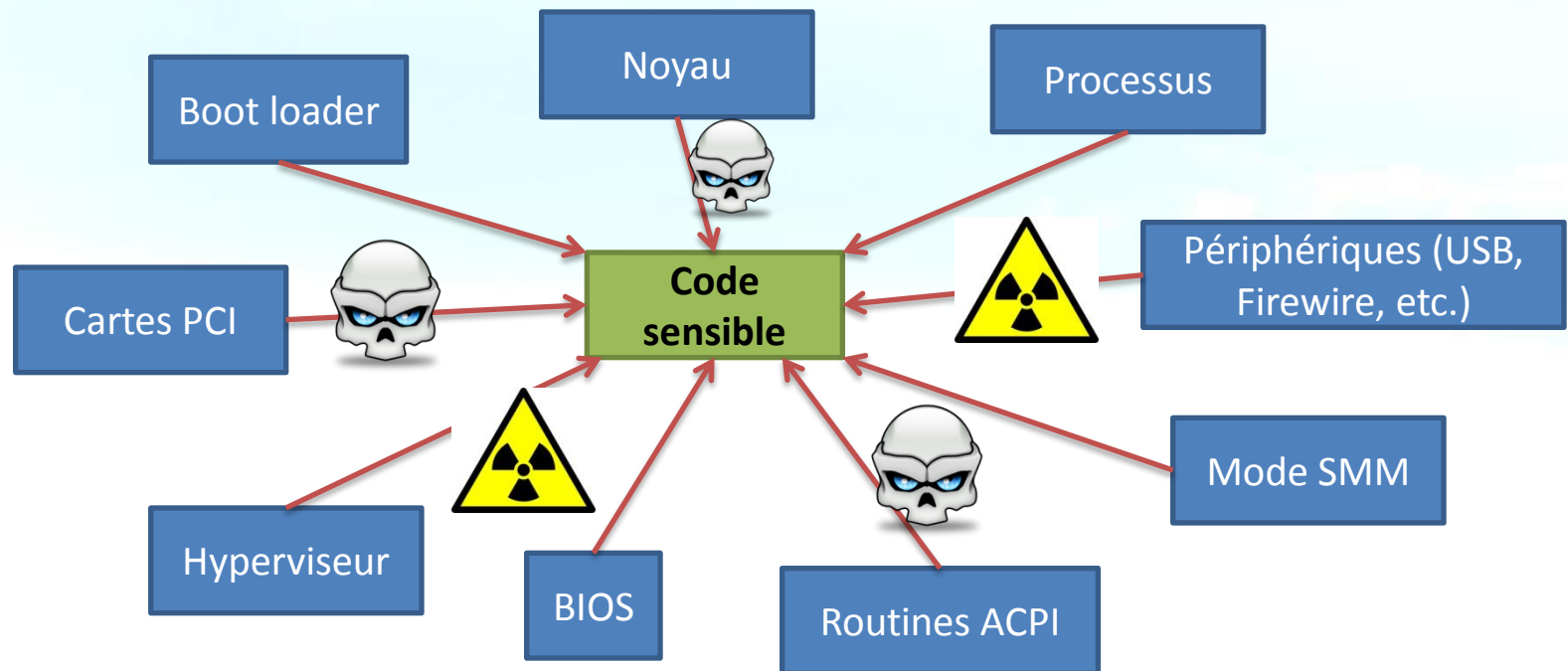
Bilan sur les usages

- Peu de produits commerciaux
 - MS Bitlocker, Sirrix AG, etc.
- Peu de produits réellement déployés
 - Alors que 300 millions de TPM sont dans la nature
- Les cas d'usages les plus courant sont le chiffrement conditionnel et les trousseaux de clés sécurisés...
- ... alors que d'autres usages mériteraient d'être mis en avant (vérification d'intégrité)

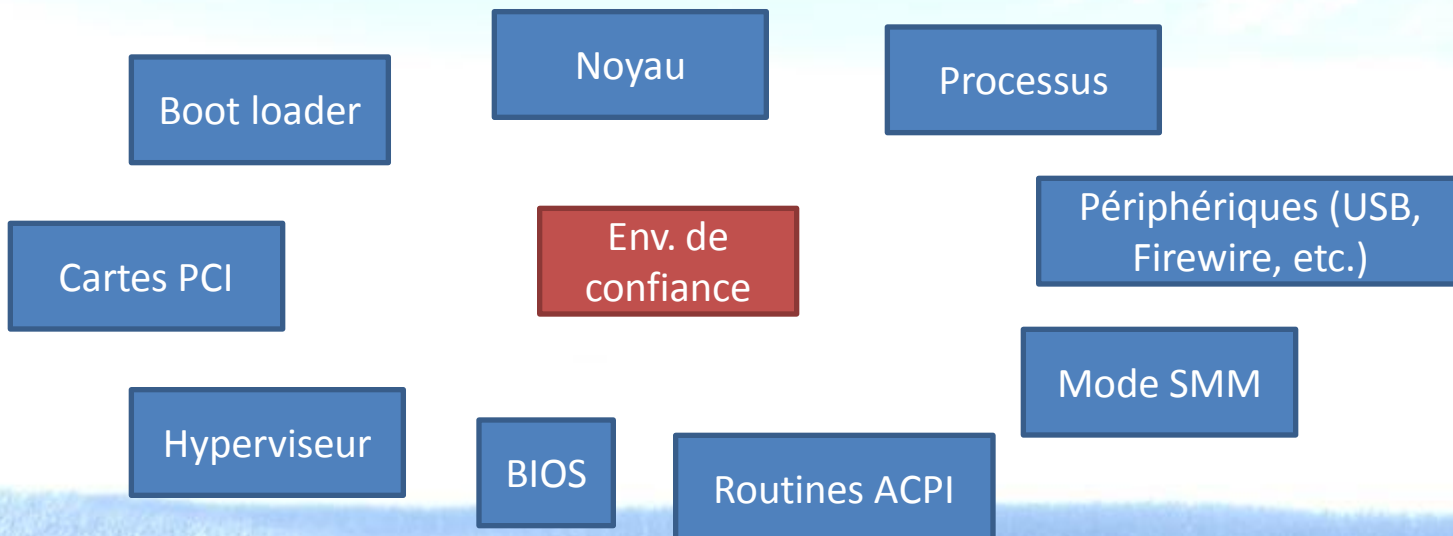


Intel TXT

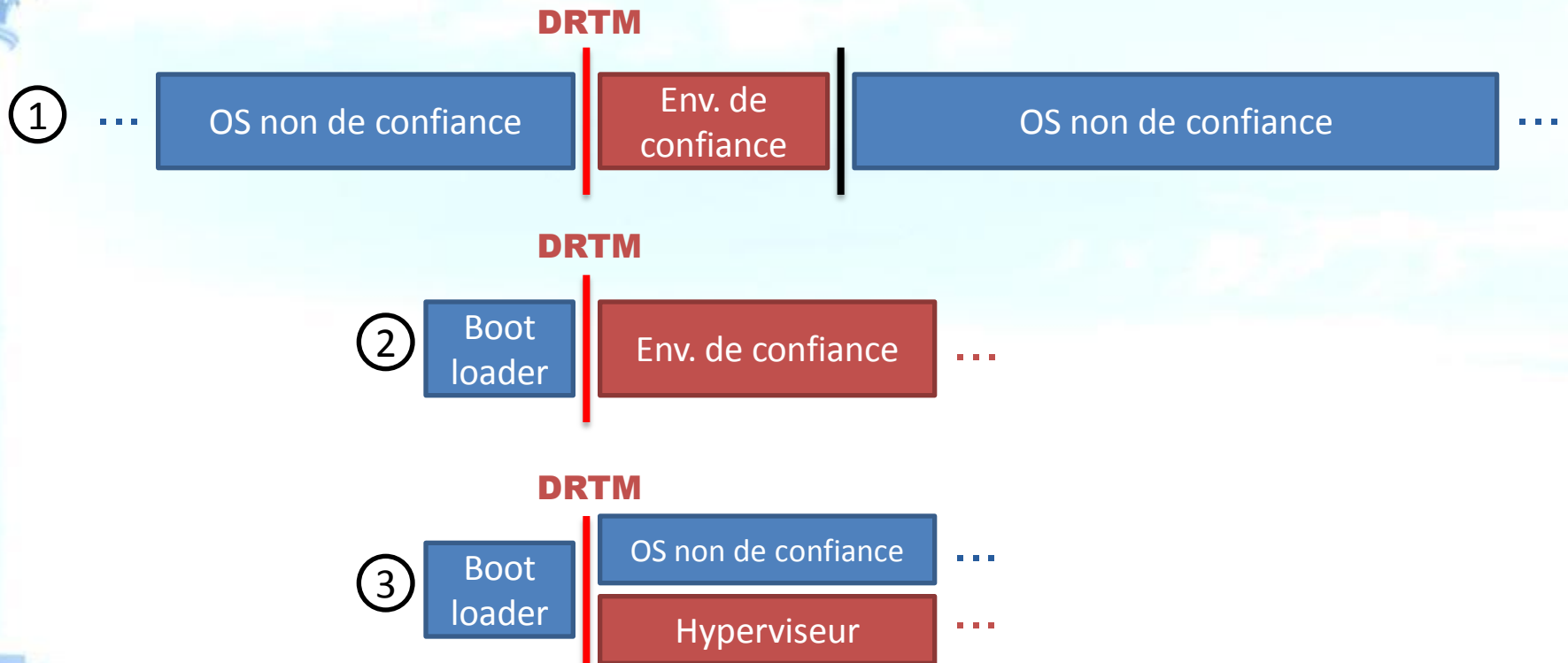
- Comment protéger l'exécution d'un code sensible dans environnement non de confiance ?



- Objectifs
 - Démarrage à chaud (*Dynamic Launch*) d'un environnement de confiance
 - Protections mémoire de l'environnement, afin d'éviter toute compromission depuis l'extérieur
 - Vérification d'intégrité au démarrage de l'environnement vis-à-vis d'une politique de sécurité (optionnel)



- Exemples de cas d'usages d'Intel TXT





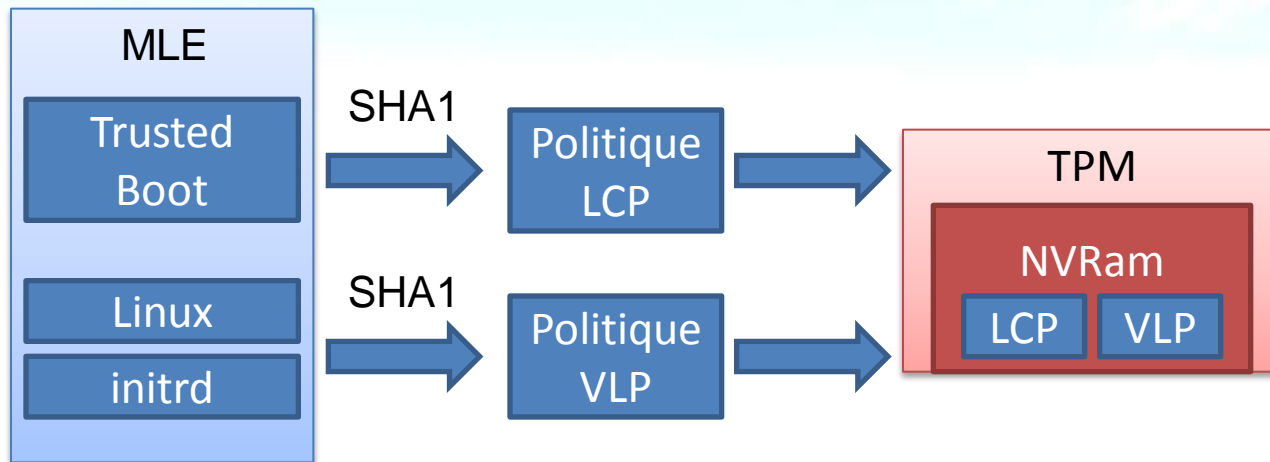
- Technologies sous-jacentes
 - Puce TPM 1.2
 - Processeur supportant la virtualisation matérielle (Intel VT-x)
 - Processeur supportant l'extension SMX (Safer Mode Extensions)
 - Chipset supportant TXT et l'IOMMU (Intel VT-d)



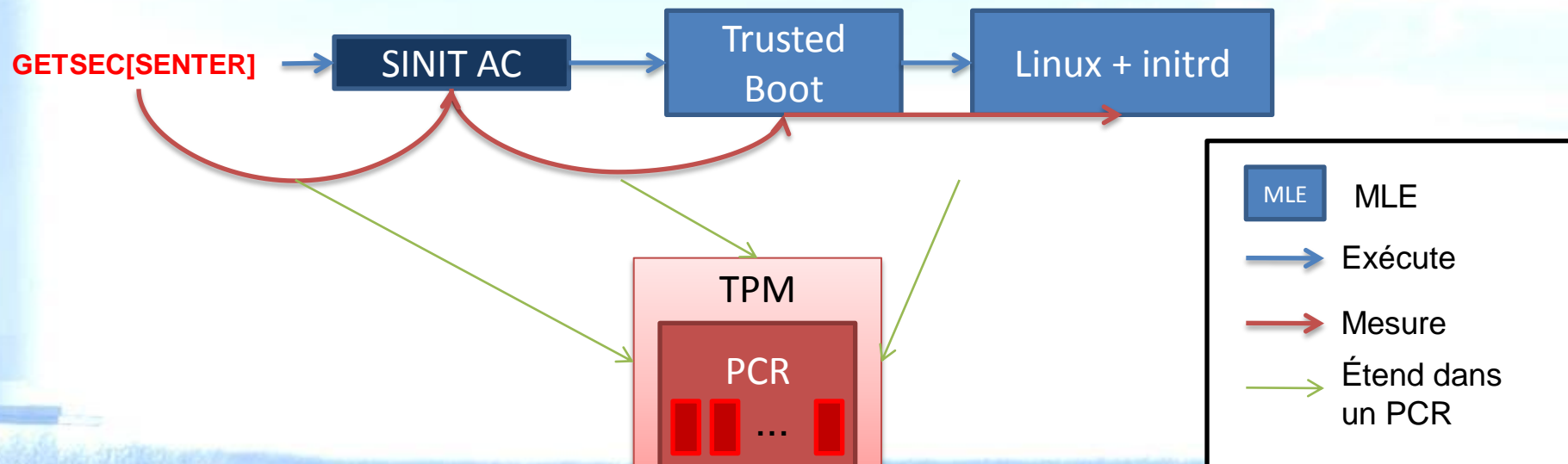
Intel TXT Trusted Boot

- Trusted Boot
 - Principale implémentation d'Intel TXT
 - Objectif : réaliser une vérification d'intégrité (*Verified Launch*) de l'OS lors de son démarrage
 - OS supportés :
 - ✓ Linux : support d'Intel TXT depuis la version 2.6.32
 - ✓ Xen (depuis la 3.2)

- Définition des politiques de sécurité
 - Intégrité d'un état connu sauvegardée dans 2 politiques de sécurité
 - ✓ LCP : *Launch Control Policy*
 - ✓ VLP : *Verified Launch Policy*
 - LCP et VLP stockées de manière sécurisée dans le TPM par le propriétaire



- Processus de mesure d'intégrité de l'environnement MLE
 - Séquence initiée par l'instruction GETSEC[SENDER] du jeu d'instructions SMX
 - Mesure puis exécution du code SINIT AC
 - Mesure puis exécution du Secure Loader (Trusted Boot)
 - Mesure puis exécution de Linux





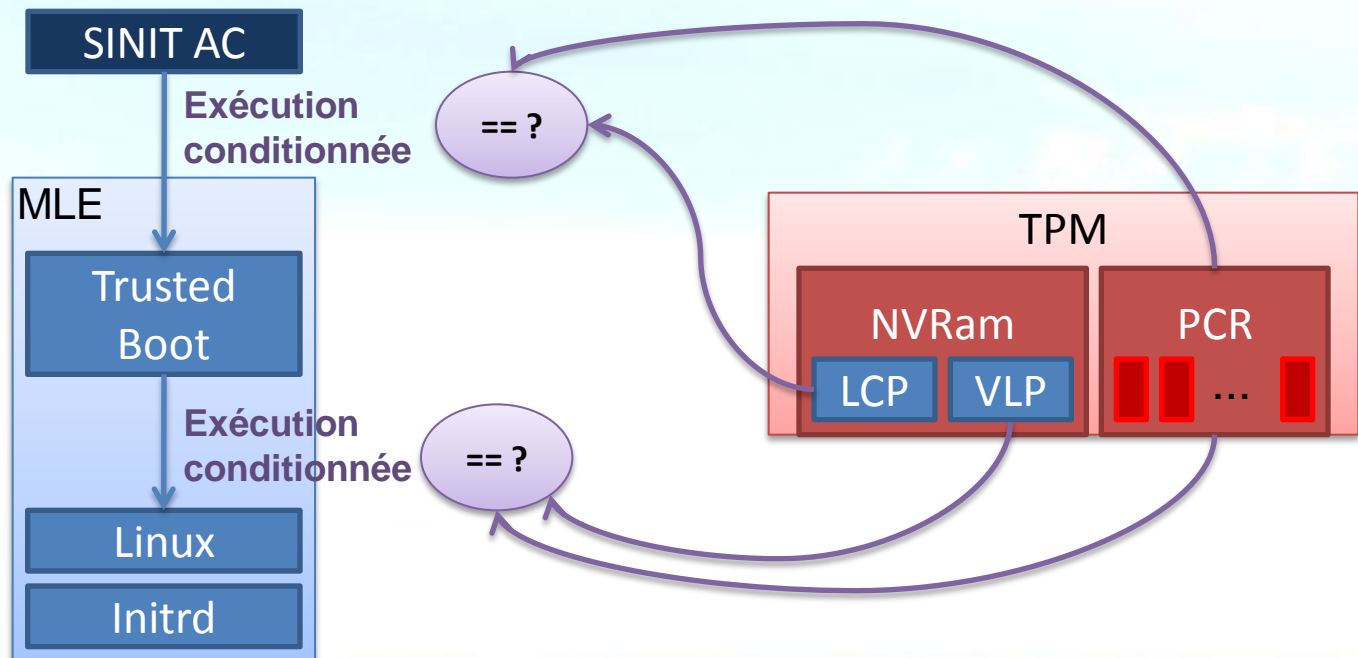
Intel TXT

Exemple de configuration de Grub

- Exemple de configuration de Grub avec Trusted Boot

```
title Trusted Linux
    root (hd0,1)
    kernel /tboot.gz
    module /vmlinuz-2.6.32 root=/dev/sda1 ro
    module /initrd-2.6.32
    module /Q45_SINIT_19.BIN
```

- Vérification d'intégrité au démarrage (*Verified Launch*)
 - Permet de conditionner le lancement de l'environnement de confiance en fonction des politiques LCP et VLP
 - Possibilité de continuer ou stopper l'exécution en cas d'intégrité non conforme





Intel TXT

Exemple de vérification d'intégrité

- Exemple de vérification avec Trusted Boot

```
TBOOT: verifying module "/boot/vmlinuz root=/dev/sda1 ro"  
TBOOT:      OK : 9e e1 ff 2f c0 92 c8 d0 39 95 2c 3c e4 a3 4c d5 da ce 5e e4  
TBOOT: verifying module "/boot/initrd.img"  
TBOOT:      OK : d7 c3 65 07 cc b2 b4 42 34 5d 8f 56 24 4a 1e cb 0e 21 3e 19  
TBOOT: all modules are verified
```

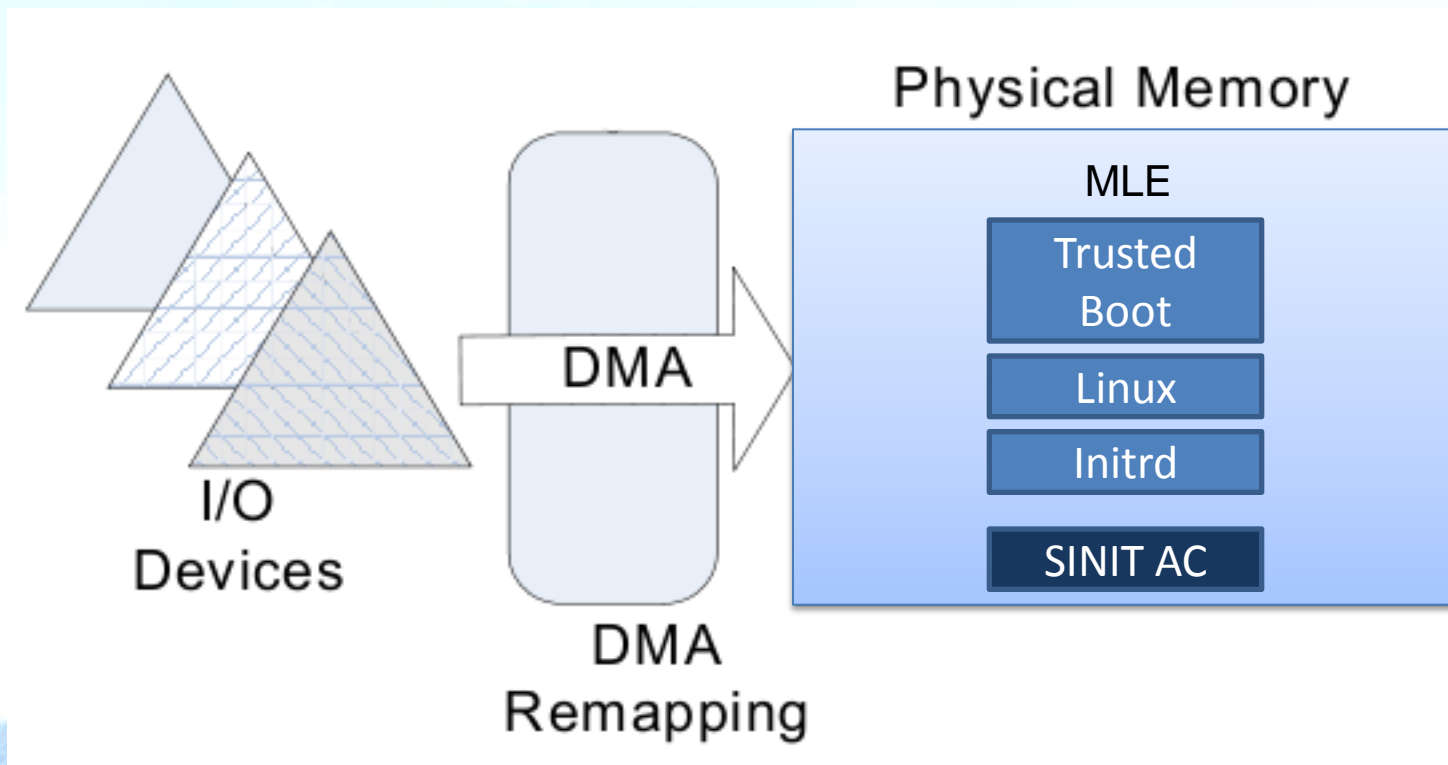


Intel TXT

Le cloisonnement

- Les protections assurées au lancement du MLE
 - Désactivation des cœurs/processeurs secondaires
 - Le code du *Secure Loader* (Trusted Boot) est chargé dans le cache du processeur avant sa mesure et son exécution
 - Désactivation des interruptions (INIT, NMI et SMI comprises)
 - Activation de la protection IOMMU contre les accès DMA
 - Blocage des mécanismes de débogage matériel
 - Communication avec le TPM sur la localité 4 (niveau de privilège maximum, non forgeable de manière logiciel)

- Protection IOMMU
 - Mise en œuvre d'une MMU dédiée aux I/O afin de bloquer l'accès DMA des périphériques et cartes PCI aux zones mémoires du MLE





Intel TXT

Le code SINIT AC

- Quelques mots sur le code SINIT AC
 - Binaire fourni par Intel, spécifique au chipset et/ou au processeur, chargé de vérifier la configuration du chipset/processeur
 - Signé numériquement par Intel (hash de la clé publique dans un registre public du chipset)
 - Disponible, pour chaque chipset, sur le site de Trusted Boot
 - Parfois directement présent dans le BIOS (ex : DQ45CB)
 - Le code du SINIT AC est chargé dans le cache ACRAM du processeur, avant d'être authentifié puis exécuté

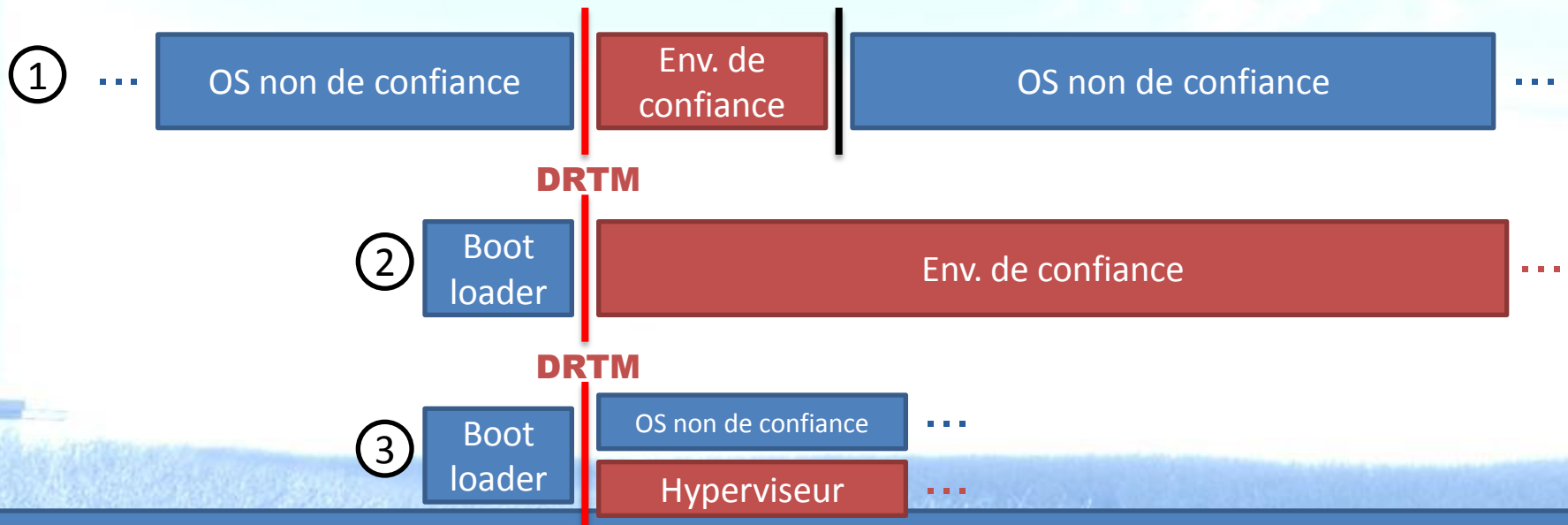


Intel TXT

Robustesse ?

- Attaques publiques contre Intel TXT
 - Vecteur ACPI (Duflot & Levillain, CanSecWest 2009 et SSTIC 2009)
 - Vecteur SMM (Duflot & Levillain, CanSecWest 2009 et SSTIC 2009 ; Wojtczuk & Rutkowska, BlackHat DC 2009)
 - Tables IOMMU (Wojtczuk & Rutkowska, Blog ITL 2009)
- Réponses possibles/proposées
 - Intel a publié un patch pour l'attaque IOMMU
 - Réaliser les opérations sensibles dans la partie totalement cloisonnée du MLE (interruptions SMI ignorées)
 - Nécessité de prendre en compte les routines SMI et ACPI dans le processus de mesure du DRTM
 - Nouveaux processeurs d'Intel (et d'AMD) incluant un moniteur de mode SMM (STM : *SMI Transfer Monitor*)

- Usages actuels d'Intel TXT
 - Essentiellement académique (projets Flicker, P-MAPS et TrustVisor)
 - Supporté dans Linux, depuis la version 2.6.32 (en association avec Tboot)
 - Supporté dans Xen (en association avec TBoot)
 - Présent dans VMWare ESXi 4.1 pour la vérification d'intégrité de l'hyperviseur au démarrage
- Types d'usages potentiellement réalisables :





- Bilan

- Technologie complexe à appréhender (nécessite de comprendre les interactions bas-niveau entre chipset/processeur/TPM)
- Technologie prometteuse, mais encore trop peu utilisée
- Malgré quelques vulnérabilités de jeunesse, TXT permet de renforcer significativement la sécurité d'un poste local :
 - ✓ Vérification d'intégrité au démarrage et/ou au *runtime*
 - ✓ Protection d'un environnement d'exécution sensible
- Possibilité de nouveaux modèles de sécurité sur architecture x86



Technologies TPM et TXT

Conclusion

- Domaine particulièrement actif
 - Intégration de plus en plus importante du TC dans des solutions libres
 - Nombreux laboratoires académiques actifs sur le TC aux Etats-Unis (CMU Cylab, MIT CSAIL, Standford) en Europe (Bochum, Graz IAIK, Cambridge) et au Japon
 - Conférences dédiées : TRUST, TIW, ATC, ETISS
 - Financements européens récurrents : FP6/OpenTC, FP7/Tecom, FP7-ICT Call 5/6, TSC Medea
 - Agences nationales impliquées : NSA (HAP), CESG, BSI et ANSSI (animation d'un groupe de travail)
 - Nouvelle version des spécifications TPM en cours d'élaboration