

garantit bien l'isolation entre les applications, quelles que soient les actions de l'utilisateur sur son propre ordinateur.

L'informatique de confiance consiste donc à garantir qu'un système informatique va se comporter de façon bien définie, y compris en présence d'attaques logicielles et/ou matérielles, le niveau de résistance à ces attaques variant en fonction des technologies utilisées.

En parallèle, une autre approche s'est développée que l'on pourrait qualifier d'informatique sécurisée. Celle-ci vise à garantir de façon forte, la confidentialité et/ou l'intégrité de programmes contre des attaques logicielles ou matérielles. De telles architectures sont proposées depuis plusieurs années (travaux de BEST [4, 6, 7], DALLAS DS5000 [10], IBM 4758 [30, 19], TRUSTNo 1 [24], XOM [26, 27, 39], AEGIS [32, 31], CRYPTOPage [23, 25, 12, 14, 11]...). Bien que leur objectif premier est différent, ces architectures peuvent être utilisées pour établir une certaine confiance dans le bon fonctionnement d'un ordinateur et présentent souvent l'avantage de fournir un niveau de résistance aux attaques, notamment physiques, plus élevé.

Dans cet article, nous présenterons un panorama des architectures permettant d'établir cette confiance. Premièrement, dans la section 2, nous aborderons les architectures dédiées à l'informatique de confiance et notamment l'architecture proposée par le TCG, puis dans la section 3, nous présenterons les principales architectures sécurisées et le niveau de confiance qu'elles peuvent apporter. Enfin, dans la section 4, nous présenterons les liens entre ces deux approches.

2 Informatique de confiance

Dans cette section nous présenterons quelques architectures dédiées à l'informatique de confiance.

2.1 Trusted Computing Group

2.1.1 Introduction

D'après son site Web [33], le *Trusted Computing Group* (TCG) est une organisation à but non lucratif formée pour développer, définir et promouvoir des standards ouverts pour du matériel permettant de l'informatique de confiance et des technologies de sécurité, pour de multiples plates-formes et périphériques. Son objectif premier est d'aider les utilisateurs à protéger leurs données sensibles (mots de passe, clés, etc.) contre une compromission due à une attaque logicielle externe ou à un vol physique.

Le TCG est divisé en plusieurs groupes de travail visant à sécuriser différents domaines de l'informatique : postes de travail, serveur, stockage, infrastructure réseau, téléphones mobiles, etc. Il a repris les spécifications développées précédemment par le *Trusted Computing Platform Alliance* (TCPA).

Le TCG regroupe plus d'une centaine d'entreprises de différents secteurs concernés par l'informatique, depuis les constructeurs (IBM, HP...) aux éditeurs de logiciels (MICROSOFT, SUN MICROSYSTEMS...) en passant par les fabricants de microprocesseurs (INTEL, AMD), de mobiles (NOKIA), de cartes à puce (GEMALTO) ou des opérateurs téléphoniques (FRANCE TÉLÉCOM, VODAPHONE...).

2.1.2 Architecture proposée

Fonctionnalités fondamentales D'après le TCG [37], une plate-forme de confiance doit fournir au minimum trois fonctionnalités de base :

fonctionnalités protégées (*protected capabilities*) : ensemble de commandes ayant seules accès à certaines zones protégées de l'architecture où les opérations sur des données sensibles peuvent être effectuées de manière sûre ;

attestation (*attestation*) : permet d'attester de la véracité d'une information ;

mesure, stockage et rapport de l'intégrité (*integrity measurement, storage and reporting*) : une architecture de confiance doit être capable d'effectuer des mesures sur les caractéristiques de la plate-forme qui sont susceptibles d'affecter son intégrité, de les stocker et de les rapporter en attestant de leur véracité.

La plate-forme de confiance Dans les systèmes TCG, il est nécessaire de faire confiance à des « racines de confiance » (*roots of trust*) car un mauvais comportement de ces dernières pourrait ne pas être détecté. Trois racines de confiance sont prévues : la racine de confiance pour la mesure (*root of trust for measurement*) capable d'effectuer des mesures d'intégrité de façon fiable, la racine de confiance pour le stockage (*root of trust for storage*) capable de maintenir l'exactitude des mesures d'intégrité, et la racine de confiance pour le rapport (*root of trust for reporting*) capable de rapporter de façon fiable les informations stockées par la racine de confiance pour le stockage. La confiance dans ces racines de confiance peut être établie par exemple par une évaluation de celles-ci par des experts.

2.1.3 Le Trusted Platform Module

Pour les ordinateurs, l'architecture de confiance proposée par le TCG est basée sur un circuit baptisé *Trusted Platform Module* (TPM [38]). Il doit pouvoir résister à la falsification. Le TCG recommande, par exemple, qu'il soit indissociable de la plate-forme à laquelle il est rattaché.

Mesure d'intégrité Le TPM intègre au minimum seize registres de configuration de plate-forme (*Platform Configuration Registers*, PCR) qui vont servir à stocker les mesures d'intégrité. L'usage exact de chaque registre et des données à mesurer est défini dans les spécifications produites pour chaque type de plate-forme par le TCG.

Au démarrage du TPM, ces registres sont mis à zéro. La commande *TPM_Extend* permet de les mettre à jour. Elle prend en paramètre le numéro du registre à modifier ainsi qu'un résumé cryptographique (calculé par le demandeur via l'algorithme SHA-1) de l'objet mesuré. Le TPM calcule et stocke dans le registre concerné le résumé cryptographique de la valeur précédente du registre concaténée avec le résumé cryptographique de l'objet. Le demandeur doit de son côté stocker dans un journal (*Stored Measurement Log*, SML) l'opération qu'il vient d'effectuer. Ainsi en comparant le journal et la valeur stockée dans un des registres, on peut s'assurer que le journal n'a pas été modifié.

Les opérations de mesure sont donc incrémentales. Dans le cas d'un PC par exemple, le code intégré au TPM va effectuer une mesure du BIOS, qui lui-même effectuera une mesure du chargeur de démarrage situé sur le disque dur, qui effectuera une mesure du système d'exploitation, etc. Un mécanisme similaire avait déjà été proposé dans le cas d'un PC classique par ARBAUGH, FARBER et SMITH [1].

Les spécifications du TCG ne définissent pas, en dehors de la phase initiale de démarrage, ce que le système d'exploitation doit mesurer et à quel moment. L'article [28] présente un exemple d'utilisation du TPM pour détecter le piratage d'un serveur Web tournant sous GNU/LINUX. Dans cet exemple, le noyau LINUX est modifié afin de mesurer les différents programmes exécutés, les modules noyau chargés, etc.

Rapport d'intégrité La racine de confiance pour le rapport d'intégrité a deux objectifs : permettre la lecture des registres contenant les mesures d'intégrité et attester de leur valeur. Le protocole de rapport d'intégrité se déroule en plusieurs étapes :

- un vérificateur demande la valeur d'un ou plusieurs PCR ;
- un agent logiciel sur la plate-forme contenant le TPM collecte les entrées du SML ;
- l'agent reçoit les valeurs des PCR depuis le TPM ;
- le TPM signe les valeurs des PCR en utilisant une AIK (*Attestation Identity Key*, un couple de clés asymétriques dédié pour l'attestation) ;
- l'agent récupère les certificats qui certifient le TPM et les envoie, ainsi que les valeurs signées des PCR et les entrées du SML au vérificateur ;
- le vérificateur recalcule, à partir du SML, les valeurs théoriques des PCR et les compare avec celles envoyées par l'agent. Il vérifie également la signature faite par le TPM et l'identité de celui-ci grâce aux certificats.

Stockage protégé La racine de confiance pour le stockage (*Root of Trust for Storage*, RTS) permet de protéger des clés et des données. Cette entité gère un petit espace de mémoire protégée où les clés sont conservées durant les opérations de signature et de déchiffrement. Les clés et données inactives peuvent être stockées en dehors du TPM afin de libérer de la place

pour d'autres. La gestion des clés lorsqu'elles sont à l'extérieur est confiée à un gestionnaire de cache de clés (*Key Cache Manager*, KCM). Ces clés sont protégées à l'aide de la clé racine pour le stockage (*Storage Root Key*, SRK), stockée à l'intérieur du TPM.

TPM v1.2 La dernière version des spécifications du TPM apporte un certain nombre de nouveautés [36] :

- Direct Anonymous Attestation : ce mécanisme, décrit dans [8], permet à l'utilisateur de prouver qu'il possède bien un TPM valide, sans avoir à révéler l'*Endorsement Key* (paire de clés unique et propre à chaque TPM) à une autorité de confiance mise en place comme dans la version précédente des spécifications.
- Localité : permet de restreindre certaines fonctions du TPM à certains acteurs en fonction de leur niveau de sécurité.
- Délégation : permet au propriétaire du TPM de déléguer le droit d'exécuter certaines fonctionnalités du TPM qui nécessitent théoriquement les droits de propriétaire à d'autres utilisateurs.
- Compteurs monotones : permet aux applications de disposer de compteurs monotones.
- Stockage et restauration de contexte : permet à plusieurs applications d'utiliser les fonctionnalités du TPM en même temps.

2.1.4 Autres composants de la vision TCG

Bien que le grand public associe généralement le TCG aux seules puces TPM qui sont de plus en plus présentes dans les ordinateurs fixes et portables, le TCG englobe toute l'infrastructure informatique : réseaux, serveurs, stockage, mobiles, etc.

Par exemple, au niveau réseau, les spécifications produites par le groupe de travail *Trusted Network Connect* [35] permettent à des opérateurs réseaux d'imposer des politiques d'accès basées sur l'intégrité (mesurée à l'aide d'un TPM ou autre) des terminaux connectés, et ainsi interdire, par exemple, la connexion au réseau de clients ne disposant pas de la bonne version du système d'exploitation.

Au niveau des téléphones mobiles, le groupe de travail mobile [34] a adapté les spécifications du TPM aux contraintes du monde mobile pour donner le MTM (*Mobile Trusted Module*). En effet, comme les différents composants d'un mobile opèrent pour des responsables différents (l'utilisateur en ce qui concerne les données, l'opérateur mobile pour la partie radio, le fabricant pour ce qui concerne la partie logiciel et processeur applicatif), le MTM prévoit plusieurs propriétaires différents empêchant notamment ainsi le propriétaire du mobile d'avoir tous les privilèges sur le MTM (contrairement à ce qui se passe pour le TPM) et ainsi de modifier, par exemple, la partie radio du mobile.

2.1.5 Critiques

De nombreuses applications pourraient bénéficier des fonctionnalités offertes par le TPM. Cependant, [16] montre que ces mêmes fonctionnalités peuvent être préjudiciables à l'écosystème informatique. Par exemple, le stockage scellé peut être utilisé pour empêcher l'interopérabilité entre des logiciels de différents fabricants. En effet, une application pourrait utiliser cette fonctionnalité pour lier un document qu'elle a créé à une configuration matérielle et/ou logicielle donnée et donc ainsi imposer qu'il ne puisse être ré-ouvert que par elle-même.

2.2 Intel Trusted Execution Technology

La *Trusted Execution Technology*, présentée par Intel dans [21] est basée sur un ensemble de composants matériels améliorés (processeur, contrôleur, TPM) conçus pour aider à protéger des informations sensibles contre des attaques logicielles. Une version préliminaire des spécifications est disponible dans [22].

2.2.1 Les objectifs

L'objectif est de fournir les services suivants :

Exécution protégée (*Protected Execution*) qui permet à des applications de tourner dans un environnement d'exécution isolé afin d'empêcher une autre application non autorisée d'observer ou de corrompre les données sur lesquelles elle travaille. Chacun de ces environnements est géré par le processeur et le noyau du système d'exploitation.

Stockage scellé (*Sealed Storage*) qui permet de protéger des clés ou des données au sein du matériel de la plate-forme en garantissant que ces dernières ne pourront être déchiffrées que si la plate-forme se trouve dans le même état que lors de leur chiffrement.

Entrées protégées (*Protected Input*) qui permettent de protéger les communications entre le clavier et la souris et l'application tournant dans un environnement d'exécution protégé.

Graphiques protégés (*Protected Graphics*) qui permettent de protéger les données envoyées par une application protégée vers l'écran contre une application espionne.

Attestation (*Attestation*) qui permet au système de prouver que l'environnement protégé a été correctement exécuté.

Lancement protégé (*Protected Launch*) qui permet de contrôler le lancement d'un système d'exploitation et d'applications critiques.

2.2.2 Le matériel

La *Trusted Execution Technology* est basée sur un certain nombre de modifications matérielles.

Au niveau du processeur, un mécanisme de partitionnement permet la création d'environnements d'exécution permettant d'isoler les applications critiques.

Au niveau du *chipset*, un mécanisme de protection mémoire, basée sur une table fournie par le processeur (*Memory Protection Table*) permet de protéger les applications qui tournent dans des partitions contre des accès mémoire effectués par d'autres applications ou via *Direct memory Access* (DMA). De plus, un système de chiffrement est intégré afin de sécuriser les communications avec le clavier, la souris et la carte graphique.

Le clavier, la souris et la carte graphique sont également modifiés afin d'intégrer un système de chiffrement permettant de sécuriser les communications.

Enfin, la *Trusted Execution Technology* nécessite la présence d'un TPM afin de permettre les fonctions d'attestation, de mesure et de stockage sécurisé.

3 Informatique sécurisée

Depuis plusieurs années, des architectures permettant de garantir la confidentialité de programmes et de leurs données et/ou leur intégrité contre des attaques logicielles et matérielles ont été proposées. On peut diviser ces architectures en deux grandes familles, celles basées sur des co-processeurs sécurisés et celles basées sur des mécanismes de chiffrement et d'intégrité de bus. Dans la suite de cette section, nous présenterons quelques unes des architectures existantes dans ces deux grandes familles.

3.1 Approche à co-processeur

Le principe de l'approche à co-processeur est de mettre tout ce qui est nécessaire à l'exécution d'un ou plusieurs programmes sensibles, c'est-à-dire un processeur, de la mémoire vive, du stockage de masse, etc., à l'intérieur d'une enceinte sécurisée résistante aux attaques matérielles. C'est par exemple le cas de l'IBM 4758 ou des cartes à puce.

3.1.1 IBM 4758

L'IBM 4758 [30, 19, 29, 20] est une carte PCI qui peut donc être insérée dans un micro-ordinateur classique. Elle contient un microprocesseur 486, de la mémoire vive, de la mémoire FLASH, de la mémoire vive sauvegardée par batterie, un générateur matériel de nombres aléatoires de qualité cryptographique, un accélérateur DES matériel, un accélérateur SHA-1 matériel, un co-processeur capable de faire de l'arithmétique modulaire sur de grands entiers (utile pour l'algorithme RSA), une horloge et une batterie. L'ensemble de ces éléments est protégé

contre des altérations physiques et contre l'espionnage par un boîtier particulier. De plus, des dispositifs de détection d'intrusion sont chargés d'effacer les secrets contenus sur la carte en cas d'attaque physique contre le boîtier.

Un système d'exploitation dédié, CP/Q++, s'exécute sur la carte. Il est chargé de faire tourner, sur le processeur de la carte, les applications stockées en mémoire FLASH et gère également les communications entre les applications et le micro-ordinateur hôte. Comme ces applications s'exécutent sur la carte, elles sont totalement protégées contre des attaques ou un espionnage depuis le monde extérieur.

Des dispositifs de sécurité sont également intégrés afin de ne permettre le chargement et l'exécution au sein de la carte que d'applications autorisées. De plus, un contrôle d'accès est mis en place pour limiter l'accès à certains secrets de la carte par les applications qui s'exécutent dessus.

3.1.2 Cartes à puce

Les cartes à puce sont très présentes de nos jours dans de nombreuses applications : banques, télécommunications, santé, transports, etc. Certaines de ces cartes ne disposent que d'une simple mémoire EEPROM mais d'autres intègrent un environnement d'exécution complet : microprocesseur, mémoire RAM et EEPROM, voire système d'exploitation et coprocesseur cryptographique, et sont capables d'héberger et d'exécuter plusieurs applications.

En fonction des applications, les cartes à puce sont conçues pour offrir un degré de résistance plus ou moins important contre différentes attaques matérielles (décapage, injection de fautes par laser, analyse de consommation et de rayonnement électromagnétique, perturbation de l'alimentation ou de l'horloge, etc.). Elles sont capables ainsi d'exécuter des applications en garantissant leur confidentialité et leur intégrité.

3.2 Approche à chiffrement de bus

L'un des problèmes des architectures à co-processeur est leur manque d'évolutivité. L'approche à chiffrement de bus consiste à considérer que seul le processeur est sécurisé et résistant contre des attaques matérielles et que tout ce qui est à l'extérieur de ce processeur (bus, mémoires, stockage, système d'exploitation, applications) peut être manipulé à volonté par un attaquant.

Afin de garantir la confidentialité et/ou l'intégrité de certains programmes, ces architectures ont recours notamment à des mécanismes de chiffrement et de protection de l'intégrité de la mémoire. Le principe de base est que toutes les informations sont chiffrées à la volée lors de leur sortie du processeur et déchiffrées lors de leur entrée dans le processeur. De même, en fonction des architectures, leur intégrité est protégée et vérifiée.

Nous allons maintenant décrire les principales architectures sécurisées proposées.

3.2.1 Best

BEST, dans [4, 5, 6, 7], pose les bases d'un microprocesseur cryptographique disposant d'un bus de données chiffré permettant d'exécuter un programme chiffré stocké en mémoire. Le programme est chiffré à l'aide d'une clé secrète, clé qui est également stockée à la construction à l'intérieur du processeur, dans un espace mémoire accessible uniquement par des mécanismes internes. Quand un bloc de données est lu depuis la mémoire, une unité de déchiffrement, intégrée au processeur le déchiffre à l'aide d'une clé dépendant de la clé secrète intégrée au processeur et de l'adresse du bloc. Quand un bloc de données est écrit en mémoire, il est au préalable chiffré de la même manière.

On peut noter que la notion d'intégrité est absente. L'attaquant ayant accès à la mémoire est capable de la modifier. La seule protection, qui ne concerne que les instructions, est la possibilité que l'attaquant, en modifiant aléatoirement un bloc, fasse exécuter au processeur une instruction d'autodestruction. De plus, ce microprocesseur cryptographique ne permet l'exécution que d'un unique programme et ne dispose d'aucun support pour un éventuel système d'exploitation.

3.2.2 Dallas DS5002FP

Les microcontrôleurs de la série DS5000 de la société *Dallas Semiconductor* intègrent des dispositifs de sécurité afin de permettre l'exécution d'un programme chiffré, et d'empêcher un attaquant contrôlant la mémoire d'avoir accès aux instructions ou aux données en clair du programme.

Par exemple, le DS5002FP [10], fabriqué depuis 1995, dispose de deux unités de chiffrement, l'une pour le bus de données et l'autre pour le bus d'adresse. Ces deux unités utilisent un algorithme de chiffrement propriétaire dépendant d'une clé secrète stockée dans un registre spécial et inaccessible au sein du microcontrôleur. L'algorithme de chiffrement du bus de données dépend également de l'adresse afin d'empêcher une attaque par permutation en mémoire. Le chiffrement des données est réalisé octet par octet.

Le chiffrement et le chargement d'un programme sont effectués par un micrologiciel (*firmware*) intégré au sein du microcontrôleur et activés via une patte spéciale. Cette opération a pour effet d'effacer la clé secrète, d'en générer une nouvelle via un générateur de nombres aléatoires interne, de charger le programme en clair via le port série du microcontrôleur, de chiffrer ce dernier avec la nouvelle clé et de stocker le programme ainsi chiffré en mémoire.

La table des vecteurs d'interruption est stockée au sein même du microcontrôleur afin qu'un attaquant ne puisse pas découvrir l'adresse chiffrée de celle-ci en déclenchant une interruption.

Le DS5002FP réalise également des accès mémoire aléatoires lorsque le bus mémoire n'est pas utilisé afin de compliquer la tâche d'un éventuel attaquant. De plus, des dispositifs de sécurité sont intégrés au sein du microcontrôleur pour permettre la destruction de la clé en cas de tentative d'altération physique.

3.2.3 ARM TrustZone

TrustZone [2, 3, 18] est une extension développée par la société ARM afin de renforcer la sécurité des applications s'exécutant sur des processeurs ARMv6 (ARM11) et vise donc principalement les applications informatiques nomades et embarquées. Cette extension impose des modifications matérielles au niveau du cœur du processeur.

Un nouveau domaine d'exécution dit sécurisé est ajouté de façon orthogonale à la séparation déjà existante entre le mode d'exécution privilégié et le mode d'exécution utilisateur. Pour rentrer dans ce nouveau domaine, le système d'exploitation doit exécuter l'instruction *Secure Monitor Interrupt*. Un moniteur sécurisé s'exécute alors et est chargé de sauvegarder le contexte d'exécution actuelle (non sécurisé) pour le restaurer ultérieurement. Il passe ensuite la main à un noyau sécurisé chargé d'exécuter les applications sécurisées.

Les lignes de cache sont marquées avec un bit indiquant si elles sont accessibles uniquement en mode sécurisé ou pas. Les autres composants internes au cœur ont également accès à cette information afin de décider si un programme a le droit d'accéder à des données.

Le contrôleur d'interruption est également modifié et deux listes de vecteurs d'interruption cohabitent, une pour traiter les interruptions se déclenchant pendant une exécution dans le domaine non sécurisé et l'autre pour celles se déclenchant pendant une exécution dans le domaine sécurisé.

3.2.4 TrustNo 1

KUHN [24] propose un microprocesseur sécurisé, baptisé TRUSTNo 1 visant à résister à des attaques matérielles (l'attaquant est supposé avoir un accès complet au matériel, excepté le processeur lui-même) ou logicielles (un système d'exploitation sous le contrôle d'un attaquant ou contenant des erreurs). L'objectif initial est de protéger certains logiciels importants contre la copie et contre des tentatives de modification de leur comportement en les chiffrant. Contrairement aux processeurs de BEST, TRUSTNo 1 apporte le support d'un système d'exploitation et permet d'exécuter plusieurs processus sécurisés grâce à un mécanisme de sauvegarde du contexte matériel du processus sécurisé lors d'une interruption.

3.2.5 XOM

LIE, THEKKATH, MITCHELL et LINCOLN [26] présentent une architecture sécurisée baptisée XOM (*eXecute-Only Memory*). L'objectif de cette architecture est de garantir la confidentialité et la bonne exécution de programmes sécurisés en présence d'attaques matérielles (l'attaquant

ayant accès à tout ce qui est à l'extérieur du processeur) ou logicielles. Ils introduisent donc la garantie de l'intégrité de l'exécution des processus sécurisés grâce à un vérificateur mémoire basé sur le calcul de MAC sur les données stockées en mémoire. Néanmoins, cette solution ne protège pas contre les attaques par rejeu ¹.

3.2.6 Aegis

SUH, CLARKE, GASSEND, DIJK et DEVADAS introduisent l'architecture AEGIS [32] offrant deux nouveaux environnements d'exécution : un environnement où toute tentative de falsification matérielle ou logicielle sera détectée (*tamper-evident environment*, TE), et un environnement garantissant en plus qu'un adversaire ne pourra pas obtenir d'informations sur le logiciel en observant le comportement du système ou en montant une attaque (*private and authenticated tamper-resistant environment*, PTR). Ils proposent deux variantes d'AEGIS, la première où seul le microprocesseur est digne de confiance, le reste (matériel et logiciels) étant potentiellement sous le contrôle d'un attaquant, et la seconde où le processeur et une partie du système d'exploitation sont dignes de confiance.

Les auteurs proposent également deux mécanismes de vérifications mémoire permettant de lutter contre les attaques par rejeu, le premier basé sur un arbre de hachage (ou arbre de MERKLE), et le second, sur des fonctions de hachage incrémentales sur des multi-ensembles [9]. L'utilisation d'arbres de hachage pour empêcher les attaques par rejeu a également été proposée par LAURADOX et KERYELL dans l'architecture CRYPTOPage-2 [25].

3.2.7 HIDE

Dans [40], ZHUANG, ZHANG et PANDE décrivent le problème de la fuite d'informations sur le bus d'adresse des microprocesseurs sécurisés. En effet, dans les architectures proposées jusque-là, le bus d'adresse du processeur sécurisé n'est pas modifié ou alors simplement chiffré. Il est donc possible d'observer les motifs d'accès à la mémoire et ainsi, par exemple, de voir qu'une ligne mémoire est plus utilisée qu'une autre. Les auteurs montrent qu'il est par exemple possible d'identifier certains algorithmes simplement en observant ces motifs d'accès ou d'obtenir des informations sur des données critiques si ces motifs d'accès dépendent de celles-ci, même en présence de caches sur le processeur.

Les auteurs présentent donc une architecture, nommée HIDE permettant de limiter l'impact de ces fuites d'information en permutant régulièrement l'espace d'adressage à l'intérieur de blocs mémoires.

Dans [17], GAO, YANG, CHROBAK, ZHANG, NGUYEN et LEE présentent des améliorations de l'infrastructure HIDE afin de réduire le nombre de lectures et d'écritures mémoire lors d'une permutation, de réduire le nombre de permutations et de conserver le principe de localité spatiale.

3.2.8 CryptoPage

L'architecture CRYPTOPage, initialement introduite par KERYELL [23], vise également à protéger la confidentialité et l'intégrité de processus en présence d'un attaquant qui contrôlerait tous les composants extérieurs au processeur. LAURADOX et KERYELL améliorent l'architecture afin de répondre aux attaques par rejeu contre la mémoire [25].

Ensuite DUC et KERYELL présentent une nouvelle version de l'architecture CRYPTOPage [11, 13] incluant également une protection contre les fuites d'informations via le bus d'adresse (mécanisme basé sur HIDE) tout en conservant des performances raisonnables comparé à une architecture non sécurisée (entre 5 et 10 % de pénalité).

¹Une attaque par rejeu contre la mémoire consiste à sauvegarder à un instant t la valeur contenue à un emplacement mémoire ainsi que l'éventuelle valeur d'authentification (MAC) associée, et de les replacer, au même emplacement mémoire, à un instant t' ultérieur.

4 Lien entre architectures sécurisées et informatique de confiance

Bien que leurs objectifs de sécurité soient différents, les architectures sécurisées et l'informatique de confiance sont liés.

Tout d'abord, les architectures de confiance telles que proposées actuellement, notamment par le TCG, ne permettent pas de remplir les objectifs de sécurité visés par les architectures sécurisées (c'est-à-dire assurer la confidentialité et/ou l'intégrité de processus contre des attaques matérielles et logicielles). En effet, le processeur n'étant pas modifié, il exécute toujours du code et manipule des données en clair. Au niveau logiciel, les applications ne sont pas protégées contre des trous de sécurité qui toucheraient le système d'exploitation, au niveau matériel, l'attaquant peut espionner le contenu de la mémoire ou des informations transmises sur les bus pour violer la propriété de confidentialité. De même, bien que le TCG prévoit la possibilité de vérification périodique de l'état du système, il reste possible de modifier le code ou les données manipulées par un processus et ainsi de casser la propriété d'intégrité en modifiant par exemple le fonctionnement du bus ou de la mémoire.

Dans le sens inverse, les architectures sécurisées récentes peuvent être utilisées pour remplir les propriétés proposées pour l'informatique de confiance.

En effet, elles intègrent des mécanismes d'attestation permettant de garantir qu'un résultat donné a bien été obtenu par l'exécution correcte d'un programme s'exécutant sur un processeur sécurisé donné. De plus, même si ce mécanisme fait défaut, si la confidentialité de l'application est garantie, elle peut emporter elle-même un mécanisme de signature des résultats et le simple fait que les résultats aient pu être produits prouve que l'application s'est exécutée sur une architecture sécurisée qui a garanti son intégrité. Ce mécanisme permet donc, si le destinataire du résultat fait confiance au fabricant du processeur, d'attester de la bonne exécution d'un programme.

De plus, la mesure de l'état de la plate-forme n'est plus nécessaire avec une architecture sécurisée car, contrairement à une architecture à base de TPM par exemple, la bonne exécution d'un programme sur une architecture sécurisée est garantie par construction, quel que soit l'état des composants externes au processeur (bus, mémoire, système d'exploitation, etc.).

5 Conclusions

L'informatique étant la pierre angulaire de la société de l'information actuelle, les contraintes sur son bon fonctionnement sont de plus en plus fortes. La sécurité de fonctionnement au sens large des systèmes, bien que propriété non fonctionnelle, devient primordiale. Longtemps assurée principalement par des moyens logiciels, on va vers une gestion plus matérielle de la sécurité.

Les progrès de l'informatique lors des soixante dernières années ont été portés par un développement exponentiel des capacités d'intégration (loi de MOORE). Malheureusement on n'arrive plus à utiliser cette profusion de transistors pour faire des processeurs séquentiels beaucoup plus rapides et on se dirige vers des ordinateurs plus parallèles et des rajouts de fonctions de sécurité accrue.

D'un côté, l'informatique de confiance (*Trusted computing*), principalement menée dans le monde industriel par le *Trusted Computing Group*, vise à assurer qu'un ordinateur va fonctionner de façon bien définie et de prouver à distance ce bon fonctionnement, en prenant comme hypothèse que les logiciels systèmes fonctionnent correctement et qu'il n'y a pas d'attaque au niveau matériel.

D'un autre côté, plutôt orienté recherche amont, de nombreuses architectures sécurisées, visant à garantir la confidentialité et/ou l'intégrité de processus, sont proposées depuis plusieurs années, principalement dans le monde académique. Ce sont ces architectures qui profitent le plus du nombre important de transistors disponibles dans les processeurs actuels ou futurs.

Ces deux approches ne sont pas à opposer et nous avons montré comment les architectures sécurisées peuvent remplir les objectifs qui sont fixés dans le cadre de l'informatique de confiance. On peut prédire que, si les développements suivent des objectifs scientifiques, les architectures sécurisées se développeront à moyen terme et remplaceront les architectures de confiance.

De très nombreuses applications peuvent bénéficier de ces nouvelles architectures, comme par exemple le calcul distribué de confiance.

Un premier exemple sont les grilles de calcul avec l'idée de pouvoir mutualiser des moyens de calcul. La disponibilité d'architectures sécurisées permet

- à un utilisateur d'empêcher à un attaquant sur une machine distante d'espionner ses programmes et données ou de modifier ses résultats ;
- au propriétaire d'un ordinateur distant de vérifier que le programme qui tourne est bien un programme autorisé, voire de prouver qu'un binaire correspond bien à un source donné [15].

Un second exemple sont les applications distribuées du monde de la santé. On pourrait imaginer le déplacement de morceaux de dossiers médicaux sous forme de processus sécurisés tournant sur du matériel sécurisé assurant que les entrées-sorties sont chiffrées tout comme la mémoire des processus pour éviter des fuites d'information personnelle à des personnes non autorisées.

Enfin, on pourrait concevoir des logiciels de gestion de droits numériques (DRM) libres dont les sources seraient publiées et qui permettraient d'utiliser des contenus protégés sur tous les systèmes d'exploitation et matériels existants, du moment qu'ils proposent un mode d'exécution sécurisé.

Néanmoins, si ces technologies sont utilisées de façon mal intentionnées, elles peuvent perturber l'écosystème informatique ou retirer des mains de l'utilisateur le contrôle des applications s'exécutant sur son propre ordinateur. Il faut donc seconder ces développements technologiques de réflexions éthiques indépendantes des différents groupes de pression présents.

Références

- [1] William A. Arbaugh, David J. Farber, and Jonathan M. Smith. A secure and reliable bootstrap architecture. In *IEEE Symposium on Security and Privacy*, pages 65–71. IEEE Computer Society, May 1997.
- [2] ARM. Trustzone : Integrated hardware and software security. White Paper, July 2004. <http://www.arm.com/pdfs/TZWhitepaper.pdf>.
- [3] ARM. Trustzone technology overview, March 2007. http://www.arm.com/products/esd/trustzone_home.html.
- [4] Robert M. Best. Microprocessor for executing enciphered programs. Technical Report US4168396, United States Patent, September 1979.
- [5] Robert M. Best. Preventing software piracy with crypto-microprocessors. In *IEEE Spring COMPCON '80*, pages 466–469. IEEE Computer Society, February 1980.
- [6] Robert M. Best. Crypto microprocessor for executing enciphered programs. Technical Report US4278837, United States Patent, July 1981.
- [7] Robert M. Best. Crypto microprocessor that executes enciphered programs. Technical Report US4465901, United States Patent, August 1984.
- [8] Ernie Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In *Proceedings of the 11th ACM conference on Computer and Communications Security (CCS'04)*, pages 132–145. ACM Press, October 2004.
- [9] Dwaine Clarke, Srinivas Devadas, Marten van Dijk, Blaise Gassend, and G. Edward Suh. Incremental multiset hash functions and their application to memory integrity checking. In *Advances in Cryptology - ASIACRYPT 2003 : 9th International Conference on the Theory and Application of Cryptology and Information Security*, pages 188–207, 2003.
- [10] Dallas Semiconductor. *DS5002FP Secure Microprocessor Chip*, July 2006. <http://datasheets.maxim-ic.com/en/ds/DS5002FP.pdf>.
- [11] Guillaume Duc. *Support matériel, logiciel et cryptographique pour une exécution sécurisée de processus*. PhD thesis, École Nationale Supérieure des Télécommunications de Bretagne, 2007. <http://enstb.org/~gduc/these/these.pdf>.
- [12] Guillaume Duc and Ronan Keryell. Portage de l'architecture sécurisée CRYPTOPage sur un microprocesseur x86. In *Symposium en Architecture nouvelles de machines (SYMPA'2005)*, pages 61–72, April 2005.

- [13] Guillaume Duc and Ronan Keryell. CRYPTOPage : an efficient secure architecture with memory encryption, integrity and information leakage protection. In *Proceedings of the 22th Annual Computer Security Applications Conference (ACSAC'06)*, pages 483–492. IEEE Computer Society, December 2006.
- [14] Guillaume Duc and Ronan Keryell. CRYPTOPage/HIDE : une architecture efficace combinant chiffrement, intégrité mémoire et protection contre les fuites d'informations. In *Symposium en Architecture de Machines (SYMPA'2006)*, October 2006.
- [15] Guillaume Duc and Ronan Keryell. Support architectural pour identification de programmes chiffrés dans une architecture sécurisée sans système d'exploitation de confiance. In *Symposium en Architecture de Machines (SYMPA'2008)*, février 2008.
- [16] Edward W. Felten. Understanding trusted computing : Will its benefits outweigh its drawbacks ? *IEEE Security and Privacy*, 1(3) :60–62, May 2003.
- [17] Lan Gao, Jun Yang, Marek Chrobak, Youtao Zhang, San Nguyen, and Hsien-Hsin S. Lee. A low-cost memory remapping scheme for address bus protection. In *Proceedings of the 15th international conference on Parallel Architectures and Compilation Techniques (PACT'06)*, pages 74–83. ACM Press, September 2006.
- [18] Tom R. Halfhill. ARM dons armor : Trustzone security extensions strengthen ARMv6 architecture. *Microprocessor Report*, 8/25/03-01, August 2004.
- [19] IBM PCI cryptographic coprocessor. <http://www-03.ibm.com/security/cryptocards/pci/cc/overview.shtml>, May 2007.
- [20] IBM 4764 PCI-X cryptographic coprocessor. <http://www-03.ibm.com/security/cryptocards/pci/cc/overview.shtml>, May 2008.
- [21] Intel. Intel® Trusted Execution Technology, Architectural Overview, 2003. <http://www.intel.com/technology/security/downloads/arch-overview.pdf>.
- [22] Intel. Intel® Trusted Execution Technology, Preliminary Architecture Specification, November 2006. <http://www.intel.com/technology/security/downloads/315168.htm>.
- [23] Ronan Keryell. CRYPTOPage-1 : vers la fin du piratage informatique ? In *Symposium d'Architecture (SYMPA'06)*, pages 35–44, Besançon, June 2000.
- [24] M. Kuhn. The TRUSTNo1 cryptoprocessor concept. Technical Report CS555, Purdue University, April 1997.
- [25] Cédric Lauradoux and Ronan Keryell. CRYPTOPage-2 : un processeur sécurisé contre le rejeu. In *Symposium en Architecture et Adéquation Algorithmique Architecture (SYMPAAA'2003)*, pages 314–321, La Colle sur Loup, France, October 2003.
- [26] David Lie, Chandramohan Thekkath, Mark Mitchell, Patrick Lincoln, Dan Boneh, John Mitchell, and Mark Horowitz. Architectural support for copy and tamper resistant software. In *Proceedings of the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS IX)*, pages 168–177, October 2000.
- [27] David Lie, Chandramohan A. Trekkath, and Mark Horowitz. Implementing an untrusted operating system on trusted hardware. In *Proceedings of the 9th ACM Symposium on Operating Systems Principles (SOSP'03)*, pages 178–192, October 2003.
- [28] Reiner Sailer, Xiaolan Zhang, Trent Jaeger, and Leendert van Doorn. Design and implementation of a TCG-based integrity measurement architecture. In *Proceedings of the 13th Usenix Security Symposium*, pages 223–238, August 2004.
- [29] Sean W. Smith. *Trusted Computing Platforms : Design and Applications*. Springer, 2004.
- [30] Sean W. Smith and Steve Weingart. Building a high-performance, programmable secure coprocessor. *Computer Networks*, 31(9) :831–860, April 1999.
- [31] G. Edward Suh, Dwaine Clarke, Blaise Gassend, Marten van Dijk, and Srinivas Devadas. Efficient memory integrity verification and encryption for secure processors. In *Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture*, December 2003.
- [32] G. Edward Suh, Dwaine Clarke, Blaise Gassend, Marten van Dijk, and Srinivas Devadas. AEGIS : Architecture for tamper-evident and tamper-resistant processing. In *Proceedings of the 17th International Conference on Supercomputing (ICS'03)*, pages 160–171, June 2003.

- [33] Trusted Computing Group. <http://www.trustedcomputinggroup.org>, February 2007.
- [34] Trusted Computing Group : Mobile, May 2008. <https://www.trustedcomputinggroup.org/groups/mobile/>.
- [35] Trusted Computing Group : Trusted Network Connect, May 2008. <https://www.trustedcomputinggroup.org/groups/network/>.
- [36] Trusted Computing Group. TPM v1.2 specification changes, October 2003. https://www.trustedcomputinggroup.org/groups/tpm/TPM_1_2_Changes_final.pdf.
- [37] Trusted Computing Group. TCG specification architecture overview, April 2004. https://www.trustedcomputinggroup.org/specs/IWG/TCG_1_0_Architecture_Overview.pdf.
- [38] Trusted Computing Group. Trusted platform module (TPM) main specification, part 1 : Design principles, part 2 : TPM structures, part 3 : TPM commands, March 2006. <https://www.trustedcomputinggroup.org/specs/TPM/>.
- [39] Jun Yang, Lan Gao, and Youtao Zhang. Improving memory encryption performance in secure processors. *IEEE Transactions on Computers*, 54(5) :630–640, May 2005.
- [40] Xiaotong Zhuang, Tao Zhang, and Santosh Pande. HIDE : an infrastructure for efficiently protecting information leakage on the address bus. In *Proceedings of the 11th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-XI)*, pages 72–84. ACM Press, October 2004.