C&ESAR 2008

Computer & Electronics Security Applications Rendez-vous

www.cesar-conference.fr

2 au 4 décembre 2008

Rennes - France

www.cesar-conference.fr

C&ESAR 2008
Computer & Electronics Security Applications Rendez-vous

« Informatique... de confiance ? »

« Can we trust... trusted computing ? »

Centre d'Electronique de l'Armement - La Roche Marguerite - BP 57419 - 35171 Bruz Cedex - Tél. : +33 (0)2 99 42 90 11 - Fax : +33 (0)2 99 42 91 01

orange Business Services

DGSIC

Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE
MINISTÈRE DE LA DÉFENSE

DGA

Préface

C&ESAR 2008 est la quinzième édition des journées SSI créées en 1997 par le Centre d'Électronique de l'Armement (Ministère de la Défense), en vue de réunir autour d'un sujet donné des acteurs gouvernementaux, industriels et académiques de la sécurité des systèmes d'information.

L'objectif visé est double, scientifique mais surtout didactique, en proposant à cette communauté SSI, c'est à dire à une audience qui va des chercheurs aux praticiens et aux décideurs, un tour d'horizon sur un sujet particulier du domaine de la sécurité des systèmes d'information. Ce sujet peut être abordé sous l'angle théorique ou pratique, matériel ou logiciel, mais toujours dans une optique pédagogique afin d'en faciliter la compréhension. C'est en effet cette vision didactique qui permettra aux utilisateurs de terrain de prendre connaissance des avancées théoriques ou techniques susceptibles de répondre à leurs besoins. C'est aussi cette vision qui permettra d'échanger sur les problèmes réels utiles aux travaux des chercheurs et des développeurs.

Après les premières années, cet aspect "échange" s'est développé avec une extension de la participation au-delà du ministère de la défense, en interministériel et dans l'industrie, faisant ainsi peu à peu de ces journées l'un des points de rendez-vous de la communauté SSI française. Cette maturation s'est aussi concrétisée par une évolution vers une forme plus classique de conférence de cycle annuel, rebaptisée en 2006 C&ESAR (Computer & Electronics Security Applications Rendez-vous), et le développement de partenariats (DGSIC, DCSSI, Orange...) sans lesquels cette conférence ne serait sans doute pas ce qu'elle est aujourd'hui.

Peut-on faire confiance... à l'informatique de confiance ?

Cette question en forme de boutade peut assez bien résumer le thème traité cette année.

Cette informatique sera considérée du point de vue matériel autant que logiciel, du point de vue du fournisseur comme de l'utilisateur. On commencera (première journée : "problématique et enjeux") par tenter de mieux définir la notion de confiance et ses liens parfois mal compris avec la sécurité, puis on évoquera quelques problématiques classiques. La seconde journée ("technologies") permettra d'aborder les techniques susceptibles d'apporter la confiance souhaitée, au travers de diverses implémentations logicielles ou matérielles, avec éventuellement quelques bémols... La dernière journée ("applications et technologies particulières") achèvera ce parcours des solutions possibles, en soulevant pour terminer quelques questions d'actualité (droits numériques, vote électronique...).

C&ESAR 2008 a reçu vingt propositions de communication, parmi lesquelles le comité de programme a retenu neuf articles au terme d'un rigoureux processus d'évaluation. Le programme de cette édition est complété par des conférences invitées de premier ordre.

Nous tenons à exprimer notre reconnaissance à tous les membres du comité de programme, ainsi, bien sûr, qu'aux auteurs des communications, qu'elles aient été retenues ou non.

Nous espérons que le programme que nous vous avons préparé vous satisfera pleinement, et nous vous souhaitons une excellente conférence.

Yves Correc, Président du Comité d'Organisation
Ludovic Mé, Président du Comité de Programme.

Comité d'Organisation

Pascal Chour	Services du Premier Ministre, SGDN/DCSSI
Yves Correc (Président du Comité d'Organisation)	Ministère de la Défense, DGA/CELAR
Olivier Heen (Directeur de la publication)	INRIA/IRISA
Ludovic Mé (Président du Comité de Programme)	Supélec

Comité de Programme

José Araujo	Alcatel-Lucent
Christophe Bidan	Supélec
Ciaran Bryce	INRIA/IRISA
Christophe Clavier	Gemalto
Yves Correc	Ministère de la Défense, DGA/CELAR
Olivier Courtay	Thomson R&D France
Loïc Duflot	Services du Premier Ministre, SGDN/DCSSI
Guy Gogniat	Lab-STICC/Université Bretagne Sud
Emmanuel Gureghian	Bertin Technologies
Olivier Heen	INRIA/IRISA
Ronan Keryell	Télécom Bretagne & HPC Project
Ludovic Mé	Supélec
Alain Merle	CEA/LETI
David Naccache	Univ. Paris 2 et ENS
David Pointcheval	ENS
Emmanuel Prouff	Oberthur
Patrick Radja	EADS
Frédéric Valette	Ministère de la Défense, DGA/CELAR

Site officiel : `http://www.cesar-conference.fr/`

Partenaires

- DGA
- DGSIC
- DCSSI
- Orange Business Services
- Supélec

Table des matières

Can computer security live with the human notion of trust ?

Jean-Marc Seigneur

Université de Genève CUI, 24 rue Général Dufour 1211, Genève 4 – Suisse
Jean-Marc.Seigneur@trustcomp.org

In the human world, two interacting persons rely on trust when there is uncertainty in result of their interaction. The requested person uses the level of trust in the requesting person as a mean to cope with uncertainty, to engage in an action in spite of the risk of a harmful outcome. There are many definitions of the human notion trust in a wide range of domains, with different approaches and methodologies : sociology, psychology, economics, pedagogy... These definitions may even change when the application domain changes. However, it has been convincingly argued that these divergent trust definitions can fit together. Interactions with uncertain result between entities also happen in the online world. So, it would be useful to rely on trust in the online world as well. However, the terms trust, trusted, trustworthy and the like, which appear in the traditional computer science literature, have rarely been based on these comprehensive multi-disciplinary trust models and often correspond to an implicit element of trust – a limited view of the faceted human notion of trust. This presentation introduces the field of computational models based on the human notion of trust and where they have been proposed in information technology, especially in computer security. Dr. Jean-Marc Seigneur is assistant professor at the University of Geneva after carrying out his PhD thesis at Trinity College Dublin on the EU-funded Future and Emerging Technologies (FET) SECURE project. The SECURE project built the first generic formal computational trust engine. His computational trust experience is now being applied to a specific application domain, namely mobile network provider trustworthiness, as part of the EU-funded FP7 PERIMETER project. He has also been an international expert reviewer for the EU Commission, the French ANR and the US Air Force.

Le multiniveau

Problématique, solutions et axes d'étude

David Boucart et Sébastien Gay

CELAR

Résumé On assiste actuellement à une forte croissance des besoins multiniveau aussi bien dans le monde militaire que civil. Nous présentons donc dans cet article tout d'abord le besoin utilisateur en distinguant bien la problématique d'accès à plusieurs niveaux de sécurité différents depuis un même poste de travail de celle d'échanges de données entre ces niveaux. Nous présentons ensuite pour commencer les différentes familles de solutions permettant de mutualiser les postes de travail, à savoir les postes distincts, le partage des périphériques, le partage de tout le matériel et les systèmes d'exploitation multiniveau. Puis nous présentons les méthodes de mutualisation du réseau, à savoir les réseaux distincts, le cloisonnement logique par marquage, le chiffrement réseau et le chiffrement ou marquage applicatif. Enfin nous présentons les familles de solutions d'interconnexion que sont les échanges par média amovibles, les échanges unidirectionnels type diode, les échanges bidirectionnels avec contrôle de flux et les échanges bidirectionnels avec contrôle des données par labellisation. Pour conclure, nous parlons des différentes architectures de clients et de passerelles regroupant ces fonctionnalités, et faisons un point sur les vulnérabilités résiduelles identifiées de ces solutions et sur les axes d'étude correspondants.

1 Problématique

Bien qu'étant une problématique abordée depuis longtemps, l'évolution récente des technologies des systèmes d'information militaires et civils mettent en évidence des besoins croissants dans le domaine du multiniveau.

Ainsi, on cherche actuellement à faire cohabiter dans les systèmes d'informations deux impératifs apparemment contradictoires :

- un besoin toujours plus grand d'accès et d'interconnexions entre une multitude de réseaux aux politiques de sécurité distinctes. Dans le monde militaire cela se traduira par des volontés d'accès ou d'échanges à des données de différents niveaux de sécurité Français, OTAN, Européens, ou même public ou d'organisations non gouvernementales. Dans le monde civil, cela se traduira par un besoin d'accès ou d'échanges entre différentes entreprises partenaires, ou entre une entreprise et le réseau internet par exemple.
- un besoin également toujours plus grand de cloisonnement des informations afin d'empêcher les attaques entre niveaux et les fuites d'informations résultantes.

Cette dualité entre un besoin d'ouverture important et un besoin de maîtrise des aspects sécurité de la gestion de plusieurs niveaux amène au final à l'implémentation de mécanismes ou d'architectures de sécurité parfois complexes. Ces mécanismes et architectures sont regroupés sous l'appellation générique de **multiniveau**.

Nous allons donc dans cet article tout d'abord analyser plus précisément le besoin utilisateur, puis détailler les différentes catégories de solutions de cloisonnement des postes de travail, de cloisonnement des réseaux et d'échanges interniveaux. Enfin nous verrons comment ces éléments peuvent

être mis en œuvre au sein d'architectures plus globales, les limitations connues et axes d'études actuels.

2 Besoin

2.1 Classification des besoins

L'appellation multiniveau est ainsi une appellation générique sous laquelle chacun peut placer un peu tout et n'importe quoi. Afin de bien appréhender le besoin utilisateur, il faut comprendre au final que l'on doit répondre principalement à deux cas de figure complémentaires :

- le besoin pour un utilisateur d'accéder à plusieurs niveaux de sécurité différents depuis son poste de travail. Ceci passe par la mutualisation des moyens, que ce soit au niveau du poste de travail ou des réseaux, et doit se traduire par une réduction des coûts et une amélioration de l'ergonomie du poste, notamment de l'encombrement ;
- le besoin d'échanger des informations entre des systèmes de niveaux différents, afin d'améliorer l'efficacité du traitement.

Ces deux besoins doivent de plus être couverts avec les objectifs de sécurité suivants :

- garantir la confidentialité : les informations d'un niveau "haut" ne doivent pas descendre vers un niveau "bas", ni même vers un niveau "haut" qui n'a pas le besoin d'en connaître [1]. Ceci doit être assuré par un cloisonnement efficace entre les niveaux d'accès ;
- garantir l'intégrité et la disponibilité : des informations ou des comportements d'un niveau "bas" ne doivent pas mettre en défaut l'intégrité d'un niveau "haut" [2]. Ceci doit être assuré par le contrôle strict des échanges entre niveaux différents.

2.2 Critères de choix des solutions

Bien qu'étant un objectif toujours tentant pour des décideurs, il est illusoire de vouloir proposer une architecture multiniveau "générique" tant le besoin et les vulnérabilités résiduelles découlant des solutions peuvent être divers et variés. Des critères de choix peuvent cependant guider l'architecte d'un système d'information vers la solution qui sera la mieux adaptée.

Comparaison des niveaux de sécurité à traiter. Le niveau de sécurité imposé par chacun des niveaux considérés est un premier élément de choix. Il est évident que plus l'écart en termes de niveau de sécurité entre deux niveaux donnés est important, plus les mécanismes multiniveau à mettre en place entre ces niveaux devront être robustes. Le fait que les niveaux ne soient pas toujours "hiérarchisables" peut cependant venir corser les choses. Par exemple entre deux niveaux de sécurité équivalents mais dépendant de domaines de sécurité distincts (par exemple entre le domaine France et OTAN, ou entre les domaines de sécurité de deux sociétés), il peut être problématique de définir une hiérarchie. Or le besoin en termes de maîtrise des mécanismes multiniveau n'en est pas négligeable pour autant.

Vulnérabilités résiduelles des solutions. D'un point de vue technique, le niveau de vulnérabilité résiduelle des solutions est également à prendre en compte. Toute solution souffre ainsi de vulnérabilités avérées ou potentielles plus ou moins importantes, qu'il est parfois difficile de quantifier précisément. Quel est en effet le niveau de sécurité d'un système basé sur un OS non maîtrisé

ou un processeur grand public ? En tout état de cause, les mécanismes multiniveaux pouvant vite devenir des usines à gaz, il faut s'efforcer d'identifier le plus précisément possible les limites techniques des architectures et mécanismes multiniveau afin de s'assurer d'une juste réponse au besoin de sécurité, ni trop ni trop peu.

Facteur humain. Enfin, l'aspect "acceptation" des solutions par les utilisateurs ne doit pas être négligé. Le besoin multiniveau n'est pas nouveau et des procédures d'échanges ou d'accès multiniveau plus ou moins contrôlés (échanges par clés USB, branchement de postes standards de manière alternée sur plusieurs niveaux différents, etc.) ont déjà été mises en œuvre. Les solutions proposées doivent donc répondre à ces besoins et fournir les moyens de faire de manière "propre" ce qui est parfois fait de manière anarchique. Mais il faudra veiller à ce que les contraintes d'utilisation ne soient pas trop importantes, sinon ces solutions seront rejetées ou contournées.

3 Solutions

3.1 Solutions de mutualisation des postes de travail

Les possibilités de mutualisation du poste de travail pour l'accès à différents niveaux peuvent être classées en quatre familles, comme le présente la figure 1. Ces familles combinent plusieurs briques technologiques, parmi lesquelles on peut citer : les commutateurs KVM, la virtualisation, le multiboot, le chiffrement de disque dur, le déport d'affichage, les OS durcis, les OS multiniveau...

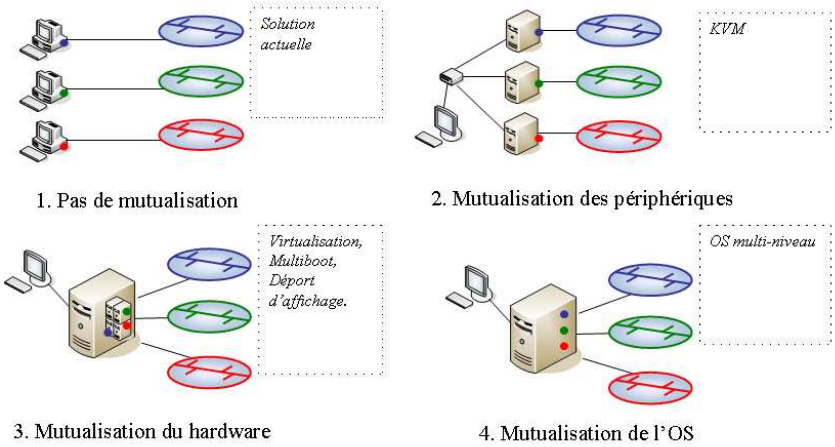


FIG. 1: Classification des solutions de mutualisation des postes.

Un poste de travail par niveau. Il s'agit de la solution actuelle, pour laquelle un poste de travail complet est dédié à chaque niveau. Ce n'est pas la solution la plus ergonomique, mais elle reste à priori celle qui apporte le plus de garantie de sécurité d'un point de vue technique, à condition de bien faire attention aux aspects Tempest.

Il faut de plus souligner que les aspects encombrement ne sont plus aussi cruciaux qu'avant, grâce au développement des PC de taille très réduite et aux écrans plats moins volumineux. Par contre, les contraintes d'usage (par exemple, la nécessité de voyager avec plusieurs portables) peuvent amener les utilisateurs à avoir des comportements à risques. Ces derniers peuvent rendre cette solution au final moins sécurisée qu'une autre.

Le partage des périphériques. Ce sont les solutions qui consistent à dédier une unité centrale complète à chaque niveau, mais à mutualiser au moins les périphériques, tels que le clavier, l'écran et la souris. D'un point de vue sécurité, ces solutions sont limitées en fonction du niveau de confiance que l'on peut avoir dans le boîtier de commutation (dit KVM pour Keyboard, Vidéo, Mouse) en lui-même, ainsi que dans les périphériques utilisés.

En première analyse, un KVM dit "mécanique" se contentant de commuter physiquement les périphériques paraît le choix approprié d'un point de vue sécurité. Cependant, ce type de KVM est très limitatif car le poste est obligé de redétecter à chaque fois les périphériques avec plus ou moins de réussite. Ces KVM s'avèrent également peu fiables, mécaniquement parlant, et présentent fréquemment des faux contacts, conduisant à des dysfonctionnements. La problématique Tempest n'est de plus pas prise en compte sur les produits proposés.

Les KVM dits "intelligents" paraissent ainsi l'option la plus viable. Ces boîtiers sont capables d'émuler les périphériques pour faciliter les transitions. Cependant, ils disposent d'une logique embarquée qui représente un niveau de complexité supplémentaire à prendre en compte par rapport à un simple KVM mécanique. Des constructeurs proposent ainsi des KVM dédiés "défense" se basant sur une architecture interne adaptée (cloisonnement des processeurs d'entrées/sorties, désactivation des fonctionnalités de commande du KVM à partir du clavier, filtrage des entrées sorties avec les périphériques, aspects Tempest).

La question des fuites d'informations via les périphériques à mémoire (souris, clavier, écran, voire périphériques USB quand le KVM le permet) reste cependant quand même à regarder de près, même si certains produits affichent des mesures de sécurité sur ce sujet grâce à l'implémentation de communications unidirectionnelles entre le poste et les périphériques utilisateurs.

Le partage de tout le matériel. Une seule unité centrale et ses périphériques associés sont ici utilisés pour accéder à tous les niveaux, mais en conservant d'une manière ou d'une autre un système d'exploitation utilisateur par niveau. L'idée est de dire qu'étant donné qu'il semble difficile d'assurer un cloisonnement multiniveau poussé au sein d'un unique OS, on va plutôt chercher à faire fonctionner de manière alternée ou simultanée des OS distincts sur un même poste tout en assurant un bon niveau d'étanchéité entre eux.

Cette catégorie de solutions est celle où les pistes de solutions sont les plus nombreuses. On distinguera dans cette famille deux sous types de solutions :

- les solutions permettant uniquement un accès alterné, nécessitant la réinitialisation du système (reboot) pour changer de niveau ;
- les solutions permettant un accès simultané à tous les niveaux, c'est-à-dire permettant à un utilisateur de passer d'un niveau à l'autre sans avoir à rebooter son poste.

Solutions à accès alterné. Les solutions à accès alterné partent du principe que certains utilisateurs n'ont en fait besoin que d'accès alternés à des informations de niveau différent, et qu'ils sont prêts à accepter un reboot de leur poste. On pourra citer par exemple un utilisateur travaillant sur un réseau d'entreprise protégé et ayant besoin d'accéder ponctuellement à Internet ou au réseau d'un partenaire, ou encore une autre personne se déplaçant et ayant besoin de faire passer son poste d'un réseau à un autre.

De nombreuses technologies permettent d'implémenter ce type de solutions :

- les systèmes de commutation "mécaniques" type commutateurs de disques durs locaux permettant de choisir de booter sur un disque ou un autre en fonction du niveau de sécurité ;
- les systèmes de cloisonnement cryptographiques des disques dur, type "Full Disk Encryption", où l'ensemble du disque dur est chiffré. Ces systèmes peuvent être matériels (puce de chiffrement intégrée dans le poste ou dans le disque) ou logiciels (logiciel de chiffrement FDE). Il est ainsi envisageable d'assurer un cloisonnement cryptographique entre deux disques ou partitions en interdisant le déchiffrement simultané aux deux disques ou partitions ;
- les mécanismes de déport de disques sur le réseau (i-SCSI, ATA over Ethernet AoE) permettant le boot sur un serveur distant. Dans ce cas on assure le cloisonnement au niveau du réseau.

Solutions à accès simultané. Les solutions à accès simultané sont plus ambitieuses mais également plus complexes. Elles vont consister à apporter un accès simultané à plusieurs OS utilisateurs sans besoin de reboot du poste. On distingue ici deux approches, selon que l'OS est exécuté localement ou sur un serveur distant.

La première approche, où l'OS est exécuté localement sur le poste, regroupe les technologies suivantes :

- Les mécanismes de virtualisation, qui vont effectuer une partition des ressources physiques d'un poste entre plusieurs instances d'OS invités. Deux types de virtualisation sont de plus possibles, selon que le virtualiseur fonctionne directement au dessus du matériel (type I) ou est implémenté au sein d'un OS hôte privilégié (type II). D'un point de vue sécurité les virtualiseurs de type I sont ainsi à privilégier[4].
- Les mécanismes de cloisonnement qui permettent de "durcir" et partitionner un OS. Assez similaires aux mécanismes de virtualisation, leur principe est de mettre en place des cages dans lesquelles les applications fonctionnent. Ainsi on a un noyau commun mais des contextes d'exécution applicatifs distincts. Ce type de mécanismes permet à peu près les mêmes choses que les mécanismes de virtualisation, mais avec un niveau de confiance généralement moindre.

La deuxième approche regroupe les technologies de déport d'affichage. Seul l'affichage est géré en local, l'OS et les applications étant exécutées sur un serveur distant. Un accès multiniveau est ainsi obtenu en utilisant plusieurs déports sur des serveurs de niveaux différents, la maîtrise des aspects multiniveau étant obtenue via le cloisonnement du réseau et le niveau d'étanchéité mis en place au niveau du poste local entre les sessions d'affichage.

Pour ces deux approches, on s'appuiera habituellement sur une "zone de confiance" consistant en un OS minimal durci, implémentant les mécanismes de sécurité, et non accessible à l'utilisateur lambda qui lui n'a accès qu'aux instances d'OS standards.

Un seul système d'exploitation. Cette famille de solutions est basée sur un système d'exploitation multiniveau. Ces systèmes implémentent une politique de sécurité multiniveau MLS (Multilevel Security [3]) et gèrent les aspects multiniveau via des mécanismes internes (contrôle d'accès MAC,

labellisation des objets du système, cloisonnement réseau, cloisonnement du système de fichiers, cloisonnement de l'interface graphique par exemple).

Solutions intéressantes d'un point de vue fonctionnel, elles n'en demeurent pas moins complexes à évaluer. Les mécanismes de sécurité sont ainsi répartis dans l'ensemble de l'OS, ce qui est problématique en termes d'assurance sur leur niveau de sécurité.

3.2 Solutions de mutualisation des réseaux

Les possibilités de cloisonnement au niveau réseau peuvent également être classées en quatre familles de solutions, présentées sur la figure 2. Les technologies mises en œuvre concernent d'une part les mécanismes de VLAN, et d'autre part les mécanismes IPsec.

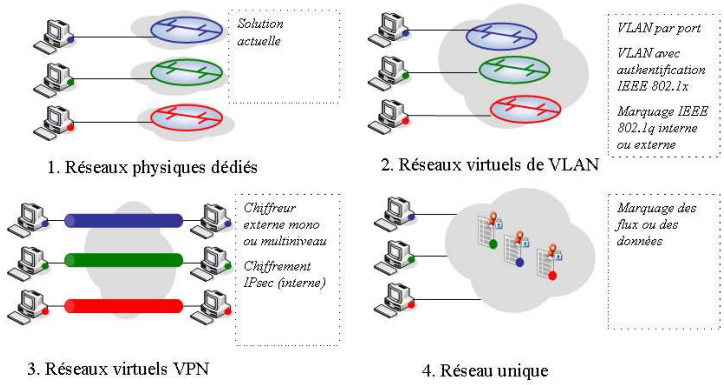


FIG. 2: Classification des solutions de mutualisation des réseaux.

Un réseau physique par niveau. C'est la solution historique, qui consiste à dédier un réseau physique à chaque niveau. Cette solution est simple mais coûte relativement cher, d'autant plus que le nombre de réseaux différents est important.

Cloisonnement logique par marquage. Cette solution consiste à assurer un cloisonnement des flux via un marquage dans lequel on aura confiance. Ce mécanisme ne protège pas en confidentialité le flux, puisque ce dernier n'est pas obligatoirement chiffré, mais a pour objectif d'assurer un cloisonnement de confiance entre les flux afin qu'ils ne se mélangent pas. Le réseau doit donc de toute manière toujours être protégé au niveau de sécurité le plus haut. L'intérêt est ici de proposer des mécanismes plus simples à implémenter que des mécanismes cryptographiques. Deux types de cloisonnement sont possibles :

- Un cloisonnement niveau 2, où il s'agit de créer des réseaux logiques virtuels (VLAN) pour chaque niveau, sur une même infrastructure physique Ethernet. Cette solution n'est applicable

que sur un réseau d'étendue limitée. Elle est simple à déployer et potentiellement acceptable sur une zone dont le périmètre physique est suffisamment maîtrisé. Selon les cas, elle peut être appliquée de façon statique (VLAN par port), de façon dynamique alterné (VLAN attribué de manière dynamique sur authentification IEEE 802.1x [5]), ou statique simultanée (application d'un marquage IEEE 802.1q [6] pour cloisonner les flux). La troisième solution est la plus intéressante d'un point de vue sécurité et économie de câblage.

– Un cloisonnement logique niveau 3 avec la labellisation des paquets IP, par l'usage des options de sécurité IP (Commercial IP Security Option [7]).

Ces solutions ne sont viables d'un point de vue sécurité que si on s'assure de la sécurité physique du réseau. Elles ne sont donc adaptées qu'à des réseaux de taille relativement réduite (par exemple une zone sécurisée avec un ensemble de bureaux, un étage d'un bâtiment, etc.) sur lesquels des mesures de sécurité physique suffisantes sont mises en place. Les communications sortant de ces zones peuvent par la suite être chiffrées. L'idée est donc au final d'économiser les chiffreurs sur des zones maîtrisées en partant du principe que cela n'est pas pire que de câbler plusieurs fois (on est de toute manière vulnérable à des attaques physiques).

Une autre condition reste la confiance en l'apposition, le traitement et la vérification du marquage. Au niveau des postes clients, cette confiance peut être amenée soit par une apposition et vérification des labels dans la zone de confiance des postes multiniveaux qui en possèdent une (postes à base de virtualisation ou de déport d'affichage par exemple), soit par le biais de commutateurs réseaux minimalistes de confiance apposant et vérifiant les labels, et placés en coupure entre les portes clients et les prises murales (TagBox). Au niveau du réseau, la confiance dans le traitement du marquage doit être apportée par des équipements de commutation et de routage fiables et de confiance. Ceci limitera probablement les fonctionnalités disponibles, comparativement aux produits de commutation du marché.

Cloisonnement par chiffrement niveau 3. Cette solution consiste à créer des VPN chiffrés. Les solutions retenues sont basées sur l'utilisation de chiffreurs IP externes ou sur l'implémentation de piles IPsec dans les zones de confiance des clients.

A noter que les aspects multiniveau impliquent des problématiques particulières. Les chiffreurs actuels sont en effet des chiffreurs "mononiveau", ne permettant que de chiffrer un seul niveau à la fois. On se retrouve donc pour l'instant à devoir empiler les chiffreurs mononiveau, ce qui est problématique en termes de coût et d'administration.

Une solution serait le développement de chiffreurs multiniveau, c'est-à-dire disposant de plusieurs interfaces physiques rouges associées à autant de niveaux de sécurité différents, et d'une unique interface physique noire. Ce chiffreur multiniveau saurait ainsi monter les tunnels IPsec correspondant aux différentes interfaces physiques.

La solution la plus intéressante serait un chiffreur multiniveau pouvant être couplé à un mécanisme de cloisonnement par VLAN de confiance. Le chiffreur joue ainsi le rôle de commutateur de confiance de la solution précédente. Il dispose dans ce cas d'une unique interface rouge multiniveau, et est capable de router les paquets vers les bons tunnels IPsec sur son interface noire en fonction du marquage appliqué du côté rouge.

Chiffrement ou marquage au niveau applicatif. Une dernière approche peut consister à chiffrer ou marquer les informations au niveau applicatif et à ne concevoir qu'un seul réseau d'échange. Ce type de solution est par exemple utilisé quand on effectue un chiffrement hors ligne dans un

niveau haut donné, et que l'on transfère la donnée noircie sur le réseau bas que l'on va alors utiliser pour le transfert. Solutions peu pratiques, les technologies de type virtualisation ou cloisonnement peuvent cependant améliorer l'ergonomie de ces solutions en permettant le chiffrement des données dans une instance ou cage dédiée, puis en facilitant le transfert d'information vers le niveau bas.

3.3 Solutions pour les échanges inter-niveau

Les possibilités d'échanges inter-niveau peuvent également être classées en quatre familles de solutions (figure 3). Ces solutions s'appuient sur des technologiques de stockage (support amovibles), sur des équipements d'interconnexion et de contrôle de flux (diode, pare-feu), ou sur des mécanismes d'analyse ou de traitement de format de fichiers.

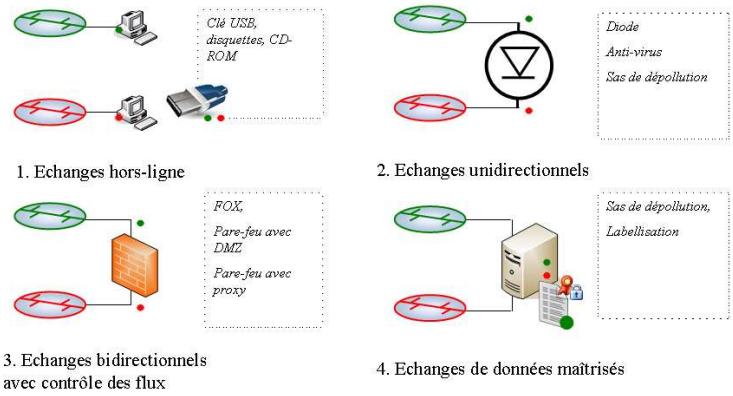


FIG. 3: classification des solutions d'échanges inter-niveau.

Les échanges "air gap". La solution traditionnelle pour échanger des informations entre des systèmes (qu'ils soient de niveaux différents ou non) demeure les supports de stockage externes, tels que les disquettes, clés USB, cd-rom... Contrairement à ce que l'on pourrait croire, cette solution est loin d'être sans risques (problèmes de fuite d'informations et d'atteintes en disponibilité) mais a l'avantage de ne pas permettre de fuite d'information automatisée à grande échelle. Elle peut être améliorée par exemple par l'utilisation d'un sas de dépollution [9], permettant de nettoyer tout document de ce qui peut présenter un risque pour le système destinataire, ou par des logiciels de contrôle des médias amovibles.

Les échanges unidirectionnels. Lorsque les échanges sont exclusivement unidirectionnels, en particulier dans le sens niveau bas vers niveau haut, l'utilisation d'un dispositif physique, appelé diode [9], ou encore d'un équipement de transfert unidirectionnel logique (par configuration d'un

pare-feu par exemple) sont des solutions actuellement disponibles. Ces équipements doivent être couplés avec des mécanismes de contrôle de contenu [8] et de vérification anti-virus pour assurer une protection en intégrité du niveau haut. Se limiter à des échanges unidirectionnels n'est cependant pas souvent acceptable d'un point de vue utilisateur.

Les échanges bidirectionnels avec contrôle des flux. Si dans le cadre d'échanges bidirectionnels les flux montants sont traités par la solution précédente, le contrôle des flux pour des échanges descendants est possible à l'aide d'architectures typiques de DMZ à base de firewalls et de proxys applicatifs. On placera couramment dans la DMZ des serveurs en coupure des flux applicatifs et on appliquera un filtrage correct des flux réseaux en interdisant les échanges directs zone interne/zone externe sans passer par la DMZ.

L'adéquation de ces solutions en termes de niveau de sécurité dépend cependant du niveau de maîtrise que l'on peut atteindre sur les flux échangés. Ainsi même si l'on s'est assuré d'un niveau de sécurité suffisant au niveau de la passerelle d'interconnexion elle-même, on ne peut pas être absolument sûr à son niveau qu'aucune fuite d'information ne se produira. Une limite évidente des mécanismes de filtrage, même les plus élaborés, reste l'utilisation de champs légitimes des protocoles filtrés pour faire fuir de l'information, à commencer par les champs destinés à contenir les données applicatives.

Les parades actuelles à ce problème consistent à s'assurer d'un bon niveau de confiance au niveau des flux par la maîtrise des systèmes émettant les informations. Ainsi couramment les flux échangés sont des flux techniques bien définis (protocoles applicatifs, flux de commande) que l'on arrive à filtrer en profondeur. Ces flux sont également générés par des applications maîtrisées et que l'on a auditées. Enfin on s'efforce de maîtriser la sécurité globale du système (flux réseau, paramétrage des équipements et machines). Ces mesures de défense en profondeur permettent au final d'utiliser ces solutions qui demeurent cependant d'un niveau de sécurité moyen, les passerelles se limitant à vérifier une syntaxe, associée à un format plus ou moins contraint, mais ne pouvant contrôler la sémantique. On limitera donc leur usage à des échanges entre niveaux de sécurité proches, ou dans un contexte déterminé et relativement restrictif quand aux données et flux échangés.

Les échanges bidirectionnels avec contrôle des informations. Pour offrir plus de fonctionnalités en termes d'échanges descendants, on peut mettre en place des mécanismes de responsabilisation des utilisateurs ou applications internes.

La solution envisagée pour faire descendre de l'information est ainsi basée sur un mécanisme de labellisation. Ce dernier consiste d'une part pour un utilisateur ou une application à autoriser un document à sortir, à le marquer, à signer le label apposé et d'autre part, au niveau de la passerelle d'interconnexion, à vérifier la présence et la validité de ce label. On part donc ainsi du principe que l'on fait confiance à l'utilisateur ou à l'application qui appose le label, mais que l'on dispose également des moyens d'engager sa responsabilité en cas de fuite d'information (signature de confiance, journalisation des échanges au niveau de la passerelle, etc.). Là où on était dans la solution précédente obligé d'assurer un niveau de sécurité correct pour l'ensemble d'un système, on se retrouve maintenant à réduire le problème à la sécurité d'une passerelle et d'un dispositif de marquage (ce qui est déjà suffisamment complexe).

On distinguera de plus pour ces fonctions deux sous cas de difficulté de réalisation croissante. Le premier plus réaliste à moyen terme est basé sur une validation humaine de chaque échange, qui pose cependant la problématique du niveau d'assurance de la visualisation du document labellisé. Comment s'assurer que ce qu'un utilisateur voit est bien ce qu'il signe ?

Le deuxième plus difficilement atteignable consiste en une labellisation automatisée effectuée par des applications devant manipuler les données. Dès qu'une application doit traiter des données de niveaux différents, et éventuellement les fusionner, il devient extrêmement difficile d'automatiser la redescente d'informations. Par exemple une agrégation d'informations non sensibles est-elle toujours non sensible ? Des cas de figure se présentent ainsi où une validation humaine restera obligatoire à moins de mettre en place des mécanismes de décision automatisés complexes et peu sûrs.

4 Conclusion

4.1 Architectures globales

Les différentes solutions présentées ci-dessus sont combinables pour fournir des architectures complètes.

Pour les aspects clients, les méthodes de mutualisation du poste et du réseau sont combinables. On peut ainsi à priori utiliser n'importe quelle méthode de cloisonnement du poste avec n'importe quelle méthode de cloisonnement du réseau.

Pour les solutions de postes clients disposant d'une zone de confiance, les mécanismes sensibles comme ceux du cloisonnement réseau peuvent y être placés (apposition de tags ou chiffrement en zone de confiance des systèmes à base de virtualisation, cloisonnement ou déport d'affichage). Pour les solutions ne disposant pas de zone de confiance, un boîtier externe sera préférable (boîtier externe pour les solutions à reboot par exemple).

Enfin, les fonctionnalités d'échanges peuvent être implémentées soit sur des serveurs du réseau, soit en local dans la zone de confiance quand c'est possible.

4.2 Vulnérabilités résiduelles identifiées et axes d'études

Fonctions à implémenter en zone de confiance. Pour ces différentes solutions, la première limitation concerne la détermination des fonctions à implémenter en zone de confiance. En effet on peut être tenté de placer un maximum de mécanismes en zone de confiance mais ce n'est pas forcément une bonne solution, puisqu'elle réduit la maîtrise que l'on peut avoir dans cette zone de confiance, et rend plus délicate une possible évaluation de ces développements. On devra donc s'efforcer d'étudier avec précision les différentes fonctionnalités à implémenter et de ne conserver en zone de confiance que ce qui est strictement nécessaire et réaliste. Par exemple pour l'implémentation de fonctions réseaux ou administration, on n'implémentera en zone de confiance que les parties de ces mécanismes impactant le plus la sécurité, le reste des développements (par exemple toutes les parties protocolaires) pouvant être repoussés vers des zones de niveau d'assurance moindre.

Dépollution et études de formats. Pour les mécanismes d'échanges, la difficulté consiste aujourd'hui à avoir un niveau de confiance convenable dans ce que l'on labellise. La faculté des formats électroniques actuels à être dynamiques (en intégrant par exemple des macros) rendent difficile l'obtention d'un bon niveau d'assurance sur cette fonction à moins de se restreindre à des formats de données très pauvres (texte ASCII) ce qui n'est pas souvent acceptable pour un utilisateur.

Sécurité matérielle. La limitation sûrement la plus importante reste le niveau de sécurité matériel des solutions utilisées. L'ensemble des mécanismes multiniveau présentés reposent en effet avant tout sur des mécanismes de sécurité matériels non maîtrisés : sécurité des processeurs, chipsets, BIOS, mémoires vives, disques durs, etc. De nombreux travaux ont récemment démontré que des failles au niveau matériel peuvent facilement remettre en cause les mécanismes de sécurité logiciels, même élaborés, reposant dessus.

Une analyse sécurité précise du fonctionnement du matériel et une multiplication des mécanismes de sécurité logiciels permettent de réduire le risque sur ces sujets. Cependant la complexité et la perpétuelle évolution des fonctions matérielles rendent cette tâche ardue et exploratoire, et multiplier les fonctions de sécurité n'est pas forcément toujours souhaitable d'un point de vue maintenabilité et ergonomie du système.

Des technologies apportent des réponses limitées à ces problèmes, comme les puces TPM permettant de se protéger des attaques en intégrité lors du boot, les mécanismes de cloisonnement des OS qui permettent de ralentir la progression d'un attaquant, l'amélioration des fonctionnalités matérielles de virtualisation (IOMMU), etc. Ces problématiques sont donc actuellement bien prises en compte par l'industrie, mais le niveau de confiance à accorder à des mécanismes à priori non maîtrisés, dépendra de l'enjeu, de la menace et de la réglementation applicable.

Références

1. D. Elliott Bell and Leonard J. La Padula : Unified exposition and multics interpretation, MTR-2997, The Mitre Corporation, July 1975
2. K. J. Biba : Integrity Considerations for Secure Computer Systems, MTR-3153, The Mitre Corporation, April 1977
3. Multilevel Security in the Department Of Defense : The Basics, 1 March 1995, `http://nsi.org/Library/Compsec/sec0.html`
4. Sébastien Gay : Apports de la virtualisation pour une plateforme informatique de confiance, C&ESAR 2008
5. IEEE 802.1X-2004 IEEE Standard for Local and metropolitan area networks – Port-Based Network Access Control, `http://standards.ieee.org/getieee802/download/802.1X-2004.pdf`
6. IEEE 802.1Q-2005 IEEE Standard for Local and Metropolitan Area Networks – Virtual Bridged Local Area Networks – Revision, `http://standards.ieee.org/getieee802/download/802.1Q-2005.pdf`
7. Commercial IP Security Option (CIPSO 2.2), IETF CIPSO Working Group, 16 July 1992, `http://tools.ietf.org/html/draft-ietf-cipso-ipsecurity-01`
8. Philippe Lagadec : Diode réseau et exefilter : 2 projets pour des interconnexions réseau hautement sécurisées, SSTIC 2006
9. Philippe Lagadec : Filtrage de messagerie et analyse de contenu, SSTIC 2004

Quelle confidentialité pour quelle confidence et avec quelle confiance ?

Gilles Trouessin

OPPIDA SUD, Le Pré Catelan – Bât. F, 78 allées Jean Jaurès, 31000 Toulouse, France
gilles.trouessin@oppida.fr

Résumé La sphère Santé-Social, notamment le soin et le médical, est un domaine où la confidentialité porte un poids particulièrement chargé en termes de confidence et en matière de confiance. Quelle confiance accorder aux Systèmes d'Information de Santé manipulant des données nominatives à caractère personnel si intimes ? Quelles solutions proposer pour permettre d'utiliser, en toute confiance et à d'autres fins que la prise en charge médico-sociale ou sanitaire, des informations confiées, à l'origine, sous couvert de la confidence singulière faite à son médecin ? L'anonymisation est efficace pour préserver la confiance tout en protégeant la confidence, si les besoins de confidentialité sont scindés entre la réversibilité – ou discrétion – traitée par chiffrement-déchiffrement des flux et l'inversibilité et/ou irréversibilité – ou séclusion – couvertes par anonymisation des données. L'anonymisation (i.e., pseudonymisation versus pseudo-anonymisation) doit présenter des caractéristiques de résistances aux risques de violation de la confidentialité liées à différents volets de la robustesse que nous décrivons et commentons dans cette présentation faite à la conférence C&ESAR, et ce afin d'apporter un nouvel éclairage sectoriel, ainsi que les éléments de base d'une démarche utile pour de telles applications se devant d'être dignes de confiance.

Mots-clés : Sphère santé-social, secteur médical-soin, suivi des trajectoires ; confidentialité, confidence, confiance ; anonymat, anonymisation, pseudo-anonymisation, pseudonymisation ; respect de la vie privée et de l'intimité, protection des données à caractère personnel ; consentement et droit à l'oubli, secret médical/professionnel et partagé/partageable ; dossier médical personnel.

Préambule : genèse de l'anonymisation en Santé-Social

Nota. Ce préambule est extrait de FOIN ou l'anonymisation du PMSI et du SNIIR-AM aparté rédigé par l'auteur pour Appariements sécurisés : statisticiens ayez de l'audace [1].

Vers le fin des années 90, la CNAMTS (Caisse Nationale de l'Assurance Maladie des Travailleurs Salariés), sur demande du Ministère du Travail et des Affaires Sociales, a confié à son entité en charge de la sécurité, le CESSI (Centre d'Etudes des Sécurité du Système d'Information), le soin de proposer une solution pour permettre la mise en œuvre fiable et sécurisée d'une version anonymisée du PMSI hospitalier (PMSI – Programme de Médicalisation du Système d'Information) pour suivre les trajectoires de soins hospitaliers et de remboursement de ces actes de soins médicaux. Rapprochant divers travaux issus de la statistique [2], de l'épidémiologie au CHU de Dijon [3], et de la recherche en cryptographie (cf. SHA-1 et schémas à seuil de Shamir), le CESSI a ainsi élaboré une démarche globale d'anonymisation [4], a rappelé ou défini des concepts de base essentiels tels que la notion de DMA - Data Matching Agency [5] et spécifié et implémenté une fonction particulière baptisée FOIN – Fonction d'Occultation d'Informations Nominatives [6]. Cette fonction FOIN, à partir du NIR[1] de

[1] NIR – Numéro d'Identification au RNIPP.
RNIPP – Registre National d'Identification des Personnes Physiques

l'assuré social ou donnant-droit (ou numéro INSEE), à partir de la date de naissance de l'ayant-droit concerné et aussi, lors des premières expérimentations, à partir du rang gémellaire de l'ayant-droit concerné, permettait, pour quiconque possédait l'ultra-secret d'anonymisation, de générer toujours et partout le même identifiant anonyme pour un même individu donné. Dans son utilisation pour le PMSI cela lui donnait ainsi la possibilité d'élaborer des statistiques nationales anonymisées sur les trajectoires de soins : ce sont les DIM (Directeurs de l'Informatique Médicale) de toutes les structures hospitalières qui devaient partager entre eux cet ultra-secret et, surtout, en préserver de concert la plus grande confidentialité. Permettant ainsi, en toute fiabilité (aucun risque de collision), en toute sécurité (aucun risque de perte de confidentialité) et en toute continuité (toujours-partout le même identifiant anonyme généré pour un même individu statistique), de remplacer des données nominatives (sinon indirectement nominatives comme le NIR) par une information strictement anonymisée par FOIN (un NIR-FOINnisé), cette fonction est applicable pour toute application du même type, à condition de partager le même ultra-secret chaque fois que l'on doit permettre des appariements, sinon d'imposer un tout autre ultra-secret si l'on n'a pas l'autorisation légale de permettre de tels appariements. Une variante de cette utilisation de FOIN a d'ailleurs été développée pour la Mairie de Paris pour son observatoire (anonymisé) des RMIstes parisiens. Tous ces travaux ont d'ailleurs ouvert la voie à une réflexion plus générale sur les risques d'inférence au-delà de la seule anonymisation [6], ainsi que sur des travaux de normalisation, tant au plan national au sein d'AFNOR [7] qu'au plan international auprès du CEN [8].

1 Introduction : du sectoriel au confidentiel

Le *respect de la confidentialité* d'une manière générale et la *protection d'un secret* en particulier font partie à la fois des bases et d'une des finalités essentielles de la *Sécurité des Systèmes d'Information (SSI)*, au sens moderne du terme. Leur mise en application (i.e., objectif de respect de la confidentialité), sinon leur mise en œuvre (i.e., exigence de protection d'un secret), ne sont pas toujours aussi simples ni aussi immédiates à réaliser qu'il n'y paraît *a priori*.

Ceci devient spécialement vrai, voire décisif, dans des environnements et contextes particulièrement sensibles, voire critiques, du point de vue d'une garantie de *confidentialité* qui est soit imposée par une obligation juridique, déontologique ou éthique, soit adossée à des situations parfois très contraignantes, techniquement, de protection du secret telles que la protection du dit (anciennement) *secret médical* et désormais dilué dans la notion beaucoup plus globale de *secret professionnel*.

La *sphère Santé-Social* en général et le *secteur du Médical / Soin / Santé* en particulier sont une illustration parfaite de la nécessité de traiter le problème de la confidentialité avec la plus grande prudence et, surtout, avec la meilleure pertinence pour répondre aux obligations juridiques et pour respecter les attentes des utilisateurs concernés.

Le fil conducteur parcouru dans cet article pour éclairer la confiance que l'on peut placer dans la protection des informations sensibles au sein de la sphère Santé-Social, souvent confiées sous le sceau de la confidence, consiste à :

- croiser les notions et perceptions habituelles que l'on a de la confidentialité avec les spécificités particulières véhiculées dans la sphère Santé-Social ;
- puis isoler les composantes juridiques, organisationnelles et techniques de cette confidentialité qui amènent à considérer l'anonymisation comme une famille de réponses permettant a priori de protéger le contexte établi de la *confidence médicale* ;

– puis décrire les caractéristiques fonctionnelles, organisationnelles et techniques de la pseudonymisation particulièrement adaptée pour instaurer la confiance dans la sphère Santé-Social.

Ainsi, à travers un parcours sectoriel des besoins / objectifs / exigences liés au respect de l'anonymat, cet article propose une nouvelle vision de la confidentialité à travers une réflexion centrée sur la confiance à accorder aux techniques d'anonymisation et de pseudonymisation utilisées dans cette sphère Santé-Social très sensible au respect de la confidence.

2 Sphère Santé-Social : exigences sectorielles particulières

La confidentialité dans le secteur médical / soin / santé est particulièrement complexe car elle implique des fondements qui touchent à *l'intimité la plus intime* de l'individu, plus intime que son secret bancaire (cf. les actifs bancaires), son secret patrimonial (cf. ses biens personnels) ou son secret à titre de citoyen (cf. liberté de pensée, opinions religieuses, politiques, syndicales, philosophiques, etc.) : il s'agit de l'intimité relative à son état de santé, aux soins qu'il reçoit et aux problèmes et/ou complications médicales dont, lui-même, n'a souvent, sinon rarement, la maîtrise. Ajoutant à cela qu'une personne malade est bien souvent très affaiblie, elle devient particulièrement vulnérable et une cible toute désignée d'atteinte à sa liberté individuelle la plus légitime, c'est-à-dire le droit au secret sur son état de santé et en ce qui concerne ses pathologies.

Sur le plan juridique, diverses lois visent à prendre en compte cette faiblesse intrinsèque pour protéger l'individu de ces nouvelles agressions sur le plan du respect de sa vie privée et de la protection de ses données à caractère personnel et à caractère médical en l'occurrence : loi de 1978 modifiée en 2004 dite loi Informatique et Liberté sur la protection des données à caractère personnel, loi de 2002 dite loi Kouchner sur le droit d'accès du patient à ses données de santé, loi de 2004 instaurant la mise en place d'un Dossier Médical Personnel (DMP) avec les précautions et sécurités qui s'imposent, etc.

Le secret ex-médical, dit secret professionnel d'ordre médical, a cette particularité qu'il est dit *d'ordre public* c'est-à-dire que le législateur a pour mission de protéger l'individu contre toute agression, y compris de le protéger contre lui-même; en d'autres termes, tout un chacun peut faire part publiquement d'informations concernant son état de santé mais son propre médecin (son médecin traitant, un généraliste ou un spécialiste, etc.), lié par ce secret médical, ne peut pas en être délié et ne peut donc pas, même sur demande de son propre patient, faire part publiquement de l'état de santé ou de détails médicaux concernant ce patient.

Ainsi, le secret d'ordre médical demande une confidentialité particulière qui est d'ailleurs traitée avec égard dans la relation particulière qu'entretient un patient avec son médecin, ou un *soigné* avec son *soignant* : il s'agit d'un dialogue particulier appelé *colloque singulier* qui, grâce à cette confidentialité particulière accordée à cette relation intime *soigné – soignant*, permet au patient de placer une confiance justifiée envers son médecin et donc de lui confier des confidences que seule cette confiance lui donne envie de confier; ce qui permet au médecin, en retour, d'instruire la plainte de son patient avec toute l'attention (i.e., diligence appropriée), l'exhaustivité (i.e., écoute de tous les symptômes exposés) et l'intégralité (i.e., soigner globalement un individu plaignant et non un organe fatigué ou malade) exigées par le bon exercice de son art, la médecine.

Il en résulte que les informations intimes, et donc confidentielles, confiées en confidence par le patient et en toute confiance au médecin, ne peuvent être réutilisées aussi facilement à d'autres desseins que celui du meilleur soin, c'est-à-dire à d'autres finalités de traitement, comme dirait la CNIL, que la meilleure prise en charge du malade. En d'autres termes, les informations médicales recueillies par le médecin ne peuvent intégralement servir aux soins (cf. intimité recluse derrière le

colloque singulier), ni intégralement aux remboursements des soins prodigués (cf. distinction entre les notions de codage des actes médicaux et de codage des pathologies), que ce soit pour l'Assurance Maladie Obligatoire (i.e., la Sécurité Sociale) au titre de la gestion du Régime Obligatoire (RO) ou pour l'Assurance Maladie Complémentaire (i.e., mutuelles-santé, assurances privées, etc.) au titre de la gestion du Régime Complémentaire (RC).

De nombreux aspects juridiques, qu'ils relèvent du légal pur ou bien du réglementaire, tels que le Code de la Santé Publique (cf. maladies soumises à déclaration obligatoire), le Code de la Sécurité Sociale (cf. utilisation du numéro dit de Sécurité Sociale), le Code de la Mutualité (cf. gestions du régime RO versus du régime RC), mais aussi et à nouveau la loi Informatique et Libertés, accordent beaucoup d'importance à faire, et faire respecter, la distinction entre la confidentialité (cf. séclusion) liée à la confidence apportée par le colloque singulier confiée en toute confiance vis-à-vis du secret médical, d'une part, et, d'autre part, la confidentialité (cf. discrétion) exigée pour toute information rendue non-médicale et utile pour procéder aux remboursements des soins ou pour les statistiques aidant à définir ou améliorer les politiques de santé publique.

Ainsi, dans la sphère Santé-Social, la confidentialité est un élément fondamental de la confiance que l'on est en droit de placer dans les Systèmes d'Information de Santé (SIS) et/ou les *Systèmes d'Information Hospitaliers (SIH)*, c'est-à-dire autour des confidences que le malade, le patient, l'assuré social (pour la Sécurité Sociale) ou l'adhérent (pour la mutuelle) veut bien confier. Il a très tôt été utile de s'essayer à faire une distinction, essentielle pour la confiance et indispensable pour la confidence, consistant à discerner entre les lieux et contextes d'utilisation –et donc les SIS/SIH associés– où l'on peut se contenter d'une confidentialité plus ou moins partageable et partagée (et donc en partie réversible) et ceux où l'on doit imposer une confidentialité unique et absolue (tellement absolue qu'elle en devient totalement irréversible) :

- dans le premier cas, on parle de confidentialité au sens de discrétion (discrétion professionnelle et/ou discrétion ordinale) : c'est par exemple le cas de l'officier de police judiciaire qui doit faire appel à un représentant de la profession (un représentant de l'ordre de médecin, en l'occurrence) pour avoir le droit d'accéder sans risque de violation du secret médical aux contenus indiscrets d'un disque dur, saisi lors d'une perquisition ou dans le cadre de l'exécution d'une procédure judiciaire ;
- dans le second cas, on parle de confidentialité au sens de séclusion (choix absolu et délibéré d'une personne à préserver sa vie privée la plus intime ou la plus recluse) : ce sont tous les cas d'utilisation de données médicales à caractère personnel, dépassant le cadre légal premier issu des finalités de traitements légitimement déclarées (études épidémiologiques, enquêtes statistiques) : hors du cadre déclaré, il faut alors pouvoir garantir l'anonymat sinon démontrer une démarche et une volonté manifeste d'anonymisation des données personnelles manipulées.

3 Confidence et confidentialité : discrétion versus séclusion

Il apparaît donc clairement et fondamentalement que tout ce qui relève de la confidentialité dans la sphère Santé-Social relève essentiellement de la confidence... d'un secret particulièrement sensible apparenté à la santé ; mais il existe plusieurs formes de secrets : tout d'abord le secret longtemps appelé secret médical au sens juridique du terme et devenu secret professionnel mais il faut aussi distinguer ce qui pourrait s'appeler à l'avenir la discrétion professionnelle.

Le secret médical ou secret professionnel relève de la déontologie en ce sens qu'il est juridiquement opposable aux médecins à travers le code de déontologie médicale et aux autres praticiens de santé à travers leurs déontologies propres : mais il n'est pas de valeur aussi probante pour toutes

ces autres professions qui ne sont pas soumises à cette obligation de secret médical, telles que les personnels administratifs des établissements de soins, les personnels de secrétariats des centres de soins ou de cabinets médicaux, les personnels internes et aussi les sous-traitants externes des services informatiques des hôpitaux publics ou des cliniques privées.

On peut alors distinguer l'obligation éthique de confidentialité au sens moral du terme (comme cela se conçoit au sujet de la présence de telle ou telle catégorie d'individus dans tel ou tel type des fichiers informatiques ou dans telle ou telle base de donnée de santé ou de soins) de l'obligation déontologique de confidentialité au sens très formel du terme (comme cela se doit au sujet de tel ou tel antécédent médical, diagnostic, symptôme ou traitement médical, ou pour telle ou telle pathologie, prescription, intervention chirurgical, etc. pour tel ou tel patient connu nominativement) : ainsi la première catégorie d'obligations relève surtout de la notion que l'on appellera ici confidentialité-discrétion, alors que la seconde conduit surtout à cette notion beaucoup plus exigeante que l'on appellera confidentialité-séclusion.

Dans des environnements moins exigeants que celui de la sphère Santé-Social et des contextes moins spécifiques que les systèmes d'information pour le soin ou le médical, ce que nous pouvons appeler confidentialité-discrétion correspondrait tout simplement à l'exigence de confidentialité au sens classique et habituel du terme. Dans ce cas, en fonction de l'expression de besoin en confidentialité et/ou de l'analyse de risque-sécurité effectuée, la protection de la confidentialité sera assurée par la mise en application habituelle des mesures classiques issues des catégories suivantes :

- **mesures juridiques :** par exemple, le respect de la loi Informatique et Libertés ou du Code de Déontologie Médicale ;
- **mesures organisationnelles :** par exemple, la définition et la révision permanente de la politique d'autorisation régissant les droits et contrôles d'accès aux données ;
- **mesures technologiques / techniques :** par exemple, la définition d'une politique d'identification-authentification adaptée, l'application d'un contrôle d'accès respectant la politique d'autorisation préétablie, l'utilisation des techniques cryptographiques les plus appropriées pour le chiffrement de flux, de messages ou de base de données, etc.

Toutes ces catégories de mesures de sécurité s'appliquent évidemment aux contextes particuliers manipulant des informations relatives à l'état de santé, aux pathologies, aux soins ou au médical pur ; elles participent à encadrer la confiance matérielle et immatérielle que la patient (ou le soigné) accorde à ces systèmes et qui font qu'il se prête au jeu de la confidence ; toutefois, il existe des environnements et des contextes où la confiance ne peut être absolue, ni la confidence intégrale :

- il s'agit, par exemple, du passage délicat depuis le cercle soin / médical vers le cercle santé / remboursement ou vers le cercle statistique / épidémiologie :
  - à l'intérieur du cercle soin / médical, il règne une obligation de discrétion (on parle de secret médical) qui applique, comme illustré supra, les mesures classiques de confidentialité, ici de confidentialité-discrétion ; ainsi ce secret particulier, appelé secret médical et devenu secret professionnel, devient aussi secret partagé entre les acteurs concernés mais il doit rester restreint au cercle des acteurs participants à la prise en charge du malade ; ceci pose l'énorme problème du dossier médical électronique ou informatisé : doit-il être partageable et/ou partagé ? Obligatoirement par essence même ! Pas nécessairement par éthique !
  - à l'extérieur de ce cercle soin / médical, par exemple lorsque l'on entre dans le cercle santé / remboursement (exemple : l'information ALD – Affection Longue Durée dans les système d'information de l'Assurance Maladie) ou dans le cercle statistique / épidémiologie, il règne une obligation de séclusion qui doit appliquer, comme il sera détaillé infra, des mesures moins classiques de protection de la confidentialité, ici de confidentialité-séclusion : c'est

d'ailleurs toute la différence entre le codage des pathologies (réservé aux soignants) et le codage maintenant appelé affiné des actes médicaux (cf. CCAM – Classification Commune de Actes Médicaux) et réservé à l'Assurance Maladie Obligatoire (au titre du RO) et plus ou moins partagé avec l'Assurance Maladie Complémentaire (au titre du RC) ;

- il s'agit encore, par exemple, du passage de la frontière entre la demande individuelle et l'attente collective :
  - la réponse médicale apportée à une demande individuelle (i.e., prodiguer les soins au patient par une prise en charge individualisée mais selon des protocoles de soins type et des mises en pratique adaptées) reste confinée dans le cercle restreint de la discrétion tout en se devant d'exporter une partie de la connaissance : jusqu'où cette exportation de connaissance depuis l'individuel vers le collectif est-elle acceptable par le patient ?
  - les réponses sanitaires élaborées pour faire face à des attentes collectives (i.e., fournir des réponses à certaines populations à risques ou face à certaines pathologies particulières, à travers des politiques globales de santé publique (nationales, régionales ou thématiques)) sont inévitablement alimentées par la connaissance des trajectoires de soins / santé, soit par accumulation de connaissances centrées sur l'individu, soit par agrégation de connaissances synthétisées sur des populations de malades déclarés ou potentiels : jusqu'où cette importation de connaissance individuelle au service du collectif est-elle exigible par l'état ?

4 Confiance et robustesses : l'anonymisation de données sensibles

Parce que la confidence (secret médical) faite dans un contexte digne de confiance (colloque singulier) ne serait plus la même dans un contexte trop élargi, et que le secret partagé ne serait plus un secret s'il était trop partagé ; et comme l'individuel (confidences du patient) doit pouvoir être au service du collectif (santé publique) de manière à ce que, en retour, le collectif (politique de santé publique) puisse aussi être au service du plus grand nombre (les plus nécessiteux, démunis et/ou affaiblis devant la maladie ; maladies les plus coûteuses, orphelines, à risques...) ; il faut pouvoir proposer des solutions dite de confiance telles que l'anonymisation pour maintenir la confidence et donc permettre d'exporter tout ou partie de la connaissance individuelle en faveur d'une exploitation collective et/ou statistique en toute confiance.

L'anonymisation est une vaste famille des solutions juridiques-organisationnelles-techniques qui permettent de rendre peu sensible (si anonymisée ou pseudonymisée) une donnée qui était hautement sensible auparavant. Beaucoup de techniques relèvent de ce que l'on pourrait appeler anonymisation, et celles qui nous intéressent ici, dans le cadre du transfert de connaissance depuis le médical vers le non-médical, ou depuis l'individuel vers le collectif, reposent dans la très grande majorité des cas sur des techniques de hachage, à base de fonctions à sens unique, qui sont censées rendre irréversible la perte de connaissance de l'identité avérée (i.e., au sens de l'état civil) correspondant à une personne physique et ce de manière non-ambigüe (sans collision) et non-équivoque (sans doublon). Beaucoup d'autres techniques de hachage ou application existent mais ne seront pas abordées ni commentées dans notre propos.

Toute la difficulté d'une démarche d'anonymisation, au sens que nous entendons ici, consiste à bâtir et à garantir la robustesse de la procédure d'anonymisation, robustesse dans diverses acceptions complémentaires du terme :

- **robustesse cryptographique :** il faut d'abord que la fonction cryptographique qui est à la base de la procédure d'anonymisation soit une réelle fonction à sens unique (MDC, MD4, MD5, SHA-1, SHA-2, etc.) ; on pourrait toujours prétendre qu'une fonction d'anonymisation à base

de chiffrement (donc réversible) serait une sorte d'anonymisation éphémère, il n'en demeure pas moins qu'elle ne peut contribuer utilement à bâtir la confiance recherchée ici pour le contexte considéré ici, à savoir anonymiser de façon irréversible pour garantir la séclusion :

- si l'on considère que l'on chiffre le nominatif (i.e., le texte clair) pour l'anonymiser en un anonymat correspondant (i.e., le cryptogramme correspondant), alors la réversibilité du chiffrement fait que l'on ne peut garantir la préservation absolue de l'anonymat, ni donc préserver la confiance dans l'anonymisation, ni donc respecter la volonté intime du patient à disposer de son droit de séclusion : ce qui fragilise sa confidence ;
- en effet, puisque tout chiffrement doit potentiellement être considéré comme mathématiquement ou cryptographiquement réversible, que l'on ait conservé ou non la clé ou bien que l'on ait pu la reconstituer ou non, que ce soit légalement par séquestre ou illégalement par cryptanalyse, le fait de disposer de la clé de déchiffrement, soit réellement et directement (lors de la génération des clés), soit virtuellement indirectement (par un moyen même hypothétique), rend peu crédible d'un point de vue éthique la propriété d'anonymat reposant sur du seul et simple chiffrement, quand bien même les politiques d'identification-et-authentification et d'autorisation-et-contrôle d'accès seraient appliquées de façon très renforcées : intrinsèquement le chiffrement se place dans un contexte de solution en réponse à une demande de discrétion et non en réponse à une demande de séclusion ;

– **robustesse fonctionnelle :** la fonction d'anonymisation doit offrir certaines bonnes propriétés fonctionnelles pour contribuer à la robustesse du système d'anonymisation et donc à la confiance à accorder à l'anonymisation :

- effet avalanche (ou différenciation) : l'effet avalanche permet de différencier clairement deux anonymats distincts (obtenus en sortie de la fonction d'anonymisation) correspondants à deux nominatifs distincts (injectés en entrée de cette fonction d'anonymisation) : cela évitera d'autant les erreurs lors de manipulations humaines ; il s'agit là d'une des bonnes propriétés d'une fonction de hachage et d'ailleurs toute la difficulté rencontrée par les recherches en cryptographie appliquée et ce qui fait que, régulièrement, il est vivement recommandé de passer à un nouvel algorithme voire à une nouvelle famille d'algorithmes (entre autres exemples : MDC puis MD4 et MD5 ; SHA (probatoire) puis SHA-1 et la série SHA-2, etc.) ;
- reproductibilité (dans l'espace) : une même procédure d'anonymisation doit être reproductible, c'est-à-dire que, à partir d'un même nominatif donné, elle doit partout générer le même anonymat correspondant, de manière à être fiable et donc de confiance : ceci contribuera aussi à éviter les collisions en interdisant de générer le même anonymat (i.e., une collision) à deux endroits différents et pour deux nominatifs distincts ;
- répétabilité (dans le temps) : une même procédure d'anonymisation doit être répétable, c'est-à-dire que, à partir d'un même nominatif donné, elle doit toujours générer le même anonymat correspondant, de manière à être fidèle et donc digne de confiance : ceci contribuera aussi à éviter les doublons en interdisant de générer des anonymats différents (i.e., des doublons) à des instants différents, pour un même nominatif donné à l'origine ;

– **robustesse technique :** la fonction d'anonymisation doit offrir des propriétés techniques pour aussi contribuer à la robustesse du système d'anonymisation et donc à la confiance à accorder à l'anonymisation elle-même :

- robustesse du hachage (pas de collisions) : pour être considéré fiable, le hachage doit, à situations égales, fournir des résultats strictement identiques et interdire les collisions ; deux nominatifs distincts doivent être anonymisés en des anonymats distincts, afin de conserver

la propriété de discrimination qui permet de discriminer entre deux nominatifs et doit aussi permettre de discriminer entre les deux anonymats générés ;

- robustesse du haché (pas de doublons) : pour être considéré fidèle, le hachage doit, en situations distinctes, fournir des résultats strictement distincts et interdire les doublons ; le même nominatif ne doit jamais être anonymisé en des anonymats distincts, afin de conserver la propriété de discrimination qui, puisqu'elle permet de reconnaître le même nominatif, doit aussi permettre de reconnaître le même anonymat correspondant ; nota : interdire les collisions et les doublons n'interdit pas d'anonymiser un même nominatif en anonymats différents et inversement, c'est-à-dire de façon différenciée pour des besoins d'anonymisation différents ; mais alors on parlera alors de fonctions ou de procédures d'anonymisation différentes, dans des contextes différents, pour des finalités différentes ; c'est tout l'intérêt de cette technique générique mais adaptable.

– **robustesse à la réversion :** pour être de confiance il faut aussi que la procédure d'anonymisation soit conçue de manière à être robustesse à la réversion, sinon elle ne garantie pas la propriété de séclusion (toujours irréversible) mais seulement une certaine forme de discrétion (parfois réversible) :
  - robustesse à la réversion directe (pas de chiffrement) : comme explicité précédemment, la fonction de base de la procédure d'anonymisation doit être irréversible, il faut donc exclure toute forme de chiffrement et, a fortiori, exclure toute autre famille de fonctions cryptographiques réversibles, pour pouvoir prétendre effectuer une anonymisation au vrai sens du terme et non pas pratiquer une sorte de pseudo-anonymisation ;
  - robustesse à la réversion indirecte (pas de table de correspondance) : plus largement, le fait de prétendre garantir l'irréversibilité impose d'interdire toute constitution de tables de correspondances entre les différents nominatifs (injectés en entrée de la fonction d'anonymisation) et les anonymats (obtenus en sortie de cette fonction) ; sinon l'observation corrélée –certes interdite mais toujours envisageable– de la listes de nominatifs à anonymiser en entrée et des anonymats obtenus en sortie permettrait de reconstituer des tables de correspondances qui, par des dichotomies-et-soustractions successives, permettent de reconstituer la bijection interdite nominatifs–anonymats ;

– **robustesse organisationnelle :** afin d'interdire toute facilité de reconstitution de cette bijection interdite reliant les nominatifs d'origine aux anonymats générés, dans la mesure où l'on aurait la possibilité de rassembler dans une même base de donnée (dite anonymisée) une quantité suffisante de données (dites à caractère personnel) et corrélées entre elles dès lors qu'elles se rapporteraient à une seule et même personne physique, il est important d'imposer certaines exigences supplémentaires en matière de robustesse organisationnelle :
  - robustesse du secret d'anonymisation : ces mesures organisationnelles couvrent la génération du secret utilisé par la fonction d'anonymisation ; il doit être considéré comme hautement confidentiel car il détermine la valeur d'initialisation de la fonction à sens unique, valeur qui doit parfois être partagée entre un grand nombre de partenaires différents, tous supposés pratiquer le même langage d'anonymisation, et la durée de vie de ce secret peut parfois être très longue car le remplacer revient à devoir générer à nouveau (i.e., avec un nouveau secret) tous les anonymats précédemment générés (i.e., avec l'ancien secret) ;
  - robustesse du chaînage : les statistiques en santé et/ou en social, les études épidémiologiques et/ou socio-économiques et autres besoins en matière d'anonymisation s'appuient souvent sur la mise à disposition de trajectoires de soins, de santé et/ou sociales ; ces trajectoires relient entre eux des épisodes de soins / santé et/ou social, à l'origine tous nominatifs mais

qu'il est indispensable d'anonymiser et, surtout, d'anonymiser de la même façon si l'on doit constituer des trajectoires cohérentes et exploitables : on parle alors de chaînage, le lien de chaînage étant généralement l'identité des individus figurant dans ces trajectoires ; la fiabilité (ou robustesse) repose sur la certitude d'anonymiser de la même façon entre tous les acteurs qui alimentent un même fichier anonymisé rassemblant des épisodes de même nature ou renseignent une même base de données répertoriant les épisodes anonymisés et chaînés en des trajectoires anonymisées ;

- menace du chaînage : le chaînage est à double tranchant car il peut être nominatif ou bien anonymisé, puisqu'il permet de corréler entre elles des informations correspondant à une seule et même personne physique pouvant être identifiée individuellement de façon nominative (avant anonymisation) ou bien pouvant être isolée individuellement mais de façon statistique (après anonymisation) ; ainsi quiconque a autorité pour anonymiser en participant à un chaînage de grande ampleur a virtuellement la possibilité de corréler tous les épisodes d'une même trajectoire (i.e., pour un individu donné), y compris les épisodes dont il n'est ni à l'origine ni le propriétaire ni l'anonymiseur ; cette visibilité longitudinale nécessite donc de prendre certaines précautions pour ne pas ouvrir la porte à des désanonymisations illégitimes (violation de la séclusion) et ne pas laisser la possibilité de ré-identifier des individus statistiques (dont l'identité avait été anonymisée) du seul fait que l'on a déjà généré leur anonymisation (transformation de l'identité nominative en une identité anonymisée) ; pour éviter cela il est nécessaire d'effectuer une multi-anonymisation (ou multi-hachage), en effectuant un hachage avant émission et dès la source (là où il est autorisé de manipuler du nominatif et d'effectuer la transposition nominatifs vers $anonymats_1$) puis un deuxième hachage après réception à la destination (là où il est imposé d'effectuer la transposition $anonymats_1$ vers $anonymats_2$) et cela avant d'entreposer les épisodes anonymisés contribuant à constituer les trajectoires anonymisées ; le cas le plus fréquemment rencontré est une anonymisation-amont (à la source des données) et une anonymisation-aval (avant exploitation des données) puis autant de ré-anonymisations que les nouvelles finalités de traitements peuvent l'imposer ;

– **robustesse à l'inférence :** pour être complet et en liaison directe avec la question générique trusting trusted computing ? posée comme fil conducteur de cette édition 2008 des journées C&ESAR et que l'on reformulera ici de la manière suivante quelle confiance pour quelle confidence en Santé-Social , il faut aussi aborder le sujet de techniques d'inférence qui, puisqu'elles permettent d'inférer des informations classifiées confidentiel à partir de données non classifiées confidentiel , doivent alors aussi permettre de retrouver des informations qualifiées nominatif à partir de séries de données qualifiées anonymisé (ou qualifiées non nominatif) et, là encore, entrer en violation de la propriété de séclusion (préserver l'anonymat) en permettant de les désanonymiser en quelque sorte :

- inférence par déduction : la corrélation (ici : le chaînage) d'informations peu sensibles en confidentialité (ici : anonymisées) peut, dans certains cas, par simple déduction (ici : logique du premier ordre, de type oui / non et et / ou), permettre d'inférer des informations plus sensibles du point de vue de la confidentialité-discrétion (ici : déduire une pathologie), voire d'inférer des informations très sensibles du point de vue de la confidentialité-séclusion (ici : déduire des identités indirectement nominatives, voire directement nominatives) ;
- inférence par induction : la corrélation d'informations peu sensibles en confidentialité (ici : anonymisées) généralisées sur un grand volume de situations comparables (ici : un ensemble de trajectoires similaires) peut, dans certains cas, par généralisation ou induction (ici :

induire, par exemple, que presque tous les patients recevant tel protocole de soins souffrent de telle pathologie), permettre d'inférer des informations plus sensibles du point de vue de la confidentialité-discrétion (ici : induire une pathologie), voire d'inférer des informations très sensibles du point de vue de la confidentialité-séclusion (ici : induire une pathologie pour tel individu désanonymisé) ;

- inférence par abduction : la corrélation d'informations peu sensibles en confidentialité (ici : anonymisées) avec des hypothèses plus ou moins hasardeuses (ici : supposer que tel anonymat est celui de telle personne) peut, dans certains cas, par supposition ou abduction (ici : l'hypthoèse d'abduction serait sachant qu'il est soigné dans tel hôpital réservé aux personnalités de l'état, supposons alors qu'il s'agit du président de la République Française), permettre d'inférer des informations plus sensibles du point de vue de la confidentialité-discrétion (ici : les soins d'une personnalité de l'état), voire d'inférer des informations très sensibles du point de vue de la confidentialité-séclusion (ici : retrouver les diagnostics, soins et états de santé successifs d'une personnalité de l'état parce que la supposition ou abduction est effectivement plausible) ;
- inférence par adduction : la corrélation d'informations peu sensibles en confidentialité (ici : anonymisées) à l'aide d'informations externes (ici : un fait divers paru dans la presse, par exemple) et observées hors du système d'information intrinsèquement considéré (ici : par exemple une base de données médicales des épisodes individuels de soins anonymisés) peut, dans certains cas, par observation ou adduction (ici : décès de tel VIP à telle date et dans tel hôpital), permettre d'inférer des informations plus sensibles du point de vue de la confidentialité-discrétion (ici : les causes du décès), voire d'inférer des informations très sensibles du point de vue de la confidentialité-séclusion (ici : retrouver les diagnostics, soins et états de santé successifs d'une personne VIP décédée et ré-identifiée avec certitude par adduction).

Tous ces cas d'exigences de robustesses ne sont pas obligatoirement, mais seulement potentiellement, applicables à tous les contextes d'anonymisation ; toutefois, selon la procédure d'anonymisation retenue, la fonction de base retenue pour l'anonymisation, les acteurs impliqués en amont et/ou en aval de la procédure d'anonymisation et dans l'exploitation des informations anonymisées (voire chaînées), la nature même des informations anonymisées (voire chaînées), les données conservées à côté des identités anonymisées, l'ampleur du chaînage d'épisodes, la nature même des épisodes chaînés, etc., etc., etc., alors il devient raisonnable de considérer que, à partir d'informations considérées non sensibles (car anonymisées), il est fort possible de retrouver des informations sensibles comme des pathologies (en violation de la confidentialité-discrétion), sinon d'inférer des informations hautement sensibles comme les identités nominatives des personnes concernées (en violation de la confidentialité-séclusion).

Comme le montre la partie haute de la figure 1, de par les bonnes questions à se poser avant de développer une solution d'anonymisation, il est déjà possible d'orienter les choix en matière d'anonymisation (réversible, irréversible, inversible) et donc en matière de solution à développer.

Comme le montre également la partie basse de la figure 1, il est également possible d'affiner l'analyse de la solution adaptée au cas de figure étudié rien qu'en analysant les contraintes de robustesse (énumérées précédemment) qui doivent s'appliquer à la solution à mettre en place.

Il n'existe pas de solution d'anonymisation universelle mais seulement des familles de solutions qu'il faut envisager les unes après les autres en fonction des réponses apportées aux questions liées aux contraintes :

- **contextuelles :** expression du besoin en anonymisation en fonction de la nature des données (secret médical, discrétion versus séclusion) ;
- **structurelles :** identification des objectifs de l'anonymisation en fonction des sources d'information et des finalités des traitements (suivi des individus, des protocoles de soins et/ou suivi des pathologies, etc.) ;
- **organisationnelles :** formalisation des exigences d'anonymisation en fonction des acteurs concernés et de rôles juridiques et organisationnels à respecter ;
- **fonctionnelles :** formalisation des exigences d'anonymisation en fonction des traitement réellement escomptés à effectuer sur les données concernées ;
- **techniques :** formalisation des exigences d'anonymisation en fonction des caractéristiques logicielles et matérielles retenues pour respecter les modalités de mise en œuvre des procédures d'anonymisation.

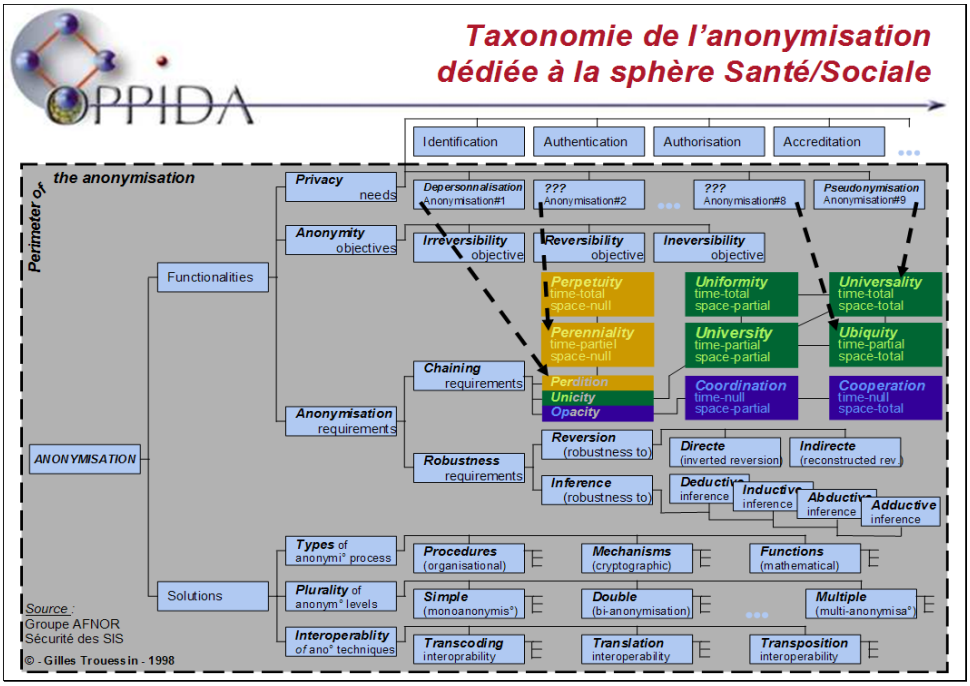


FIG. 1: Taxonomie des besoins-objectifs-exigences d'anonymisation parfaitement applicables au contexte de la sphère Santé-Social

5 Vues sectorielles en anonymisation et pseudonymisation

Tant historiquement (depuis le milieu des années 90), que qualitativement (exigence élevée de meilleure qualité des informations pouvant servir à élaborer les politiques de santé publique) ou

que quantitativement (à terme tout citoyen de la population française doit pouvoir figurer dans une base nationale, sinon dans de multiples bases régionales, locales et/ou thématiques), le premier secteur demandeur en matière d'anonymisation robuste est effectivement le secteur Santé-Social.

Ceci explique que depuis plus d'une dizaine d'années maintenant de nombreux travaux ont été menés dans le but de définir et de proposer des schémas d'anonymisation qui soient suffisamment robustes pour préserver la propriété promise et due de confidentialité-séclusion, mais également suffisamment flexibles pour permettre des utilisations multiples et variées de telles bases de données rassemblant potentiellement des informations médicales-sociales-financières individuelles et à caractère personnel et chaînées (ou chaînables), mais anonymisées avec une robustesse que l'on souhaiterait qualifier de digne de confiance .

À titre d'exemple, cela a conduit à bien identifier différentes familles de chaînage pour constituer des trajectoires reliant des épisodes (par exemple) de soin, de santé, de remboursement de soins, ou sociaux ou d'accidentologie, etc. :

- **chaînage temporel, chaînage spatial, chaînage spatio-temporel :** il s'agit de chaîner uniquement dans le temps, ou bien uniquement dans l'espace, ou bien à la fois dans le temps et dans l'espace, les différents épisodes recueillis, de manière à avoir, pour une période de temps donnée, ou pour une zone géographique donnée, ou à la fois pour une période de temps donnée et pour une zone géographique donnée, l'ensemble des épisodes concernant un individu donné : plus il y aura une connaissance exhaustive sur la trajectoire d'un individu, plus cette trajectoire risque de révéler des informations quelque peu subliminales telles qu'une pathologie (cf. risque de violation de la confidentialité-discrétion) ou risque en quelque sorte de transpirer l'identité nominative de l'individu (cf. risque de violation de la confidentialité-séclusion) ; d'où la vigilance à avoir vis-à-vis des différentes techniques d'inférence comme explicité précédemment ; par chaînage, on peut donc obtenir des trajectoires temporelles ou spatiales ou bien spatio-temporelles ;
- **chaînage thématique et spécialisé :** il s'agit là de ne chaîner que des épisodes appartenant à une thématique donnée (soit du médical pur, soit le sanitaire pur, soit le social, etc.), voire de ne chaîner, dans une thématique donnée, que les épisodes appartenant à une discipline donnée (thématique : médecine ; discipline : cancérologie) ; cette fois le risque d'inférence est intrinsèque car directement véhiculé par la thématique même de la base considérée ; par chaînage thématique, on peut donc obtenir des trajectoires par thématique voire par thématique-et-discipline ou encore par thématique-et-discipline-et-rubrique ;
- **chaînage total / partiel / nul :** il s'agit là de ne chaîner que par période de temps (couper le lien de chaînage, par exemple, tous les 1ers janvier ou tous les 30 jours ou tous les semestres) ou par zone géographique (couper le lien de chaînage, par exemple, entre les départements ou bien entre les régions administratives ou encore en répartissant les données selon les quatre quadrants de l'hexagone (nord-ouest / nord-est / sud-ouest / sud-est), de manière à diminuer la portée des trajectoires ainsi constituées et donc à réduire la connaissance sur les individus concernés ; on peut donc obtenir des trajectoires à chaînage total, ou à chaînage partiel ou encore à chaînage nul, que l'on peut croiser avec des chaînages temporels et/ou spatiaux et/ou que l'on peut aussi croiser avec des chaînages thématiques et/ou disciplines et/ou rubriques etc.

Cette variété quasi à l'infini des possibilités de chaînages est illustrée sur la figure 2, où l'on montre la fabrication des neuf types (trois fois trois) de chaînages de base (temporel-nul / temporel-partiel / temporel-total (soit trois types) croisés avec spatial-nul / spatial-partiel / spatial-total (soit trois types également)).

Lors des travaux conduits en partie sous l'égide d'AFNOR [7], chacune de ces neuf possibilités avait été baptisée de manière à pouvoir montrer la capacité de recomposition entre elles depuis le chaînage le plus pauvre (dit opacifié ou perdu ou encore unique selon que l'on parle respectivement de chaînage spatial, temporel ou spatio-temporel) jusqu'au chaînage ayant la plus forte potentialité (dit universel car toujours et partout constitué des mêmes anonymats).

Chacun de ces neuf cas de figure théoriques ne sont pas encore développés dans des applications réelles actuellement en exploitation ; en revanche certains d'entre eux sont particulièrement utiles tels que, ceux matérialisés en partie basse du schéma suivant et dits unique , universifié , uniforme , ubiquiste et universel : ils décrivent sommairement tout le spectre utile des chaînages spatio-temporels.

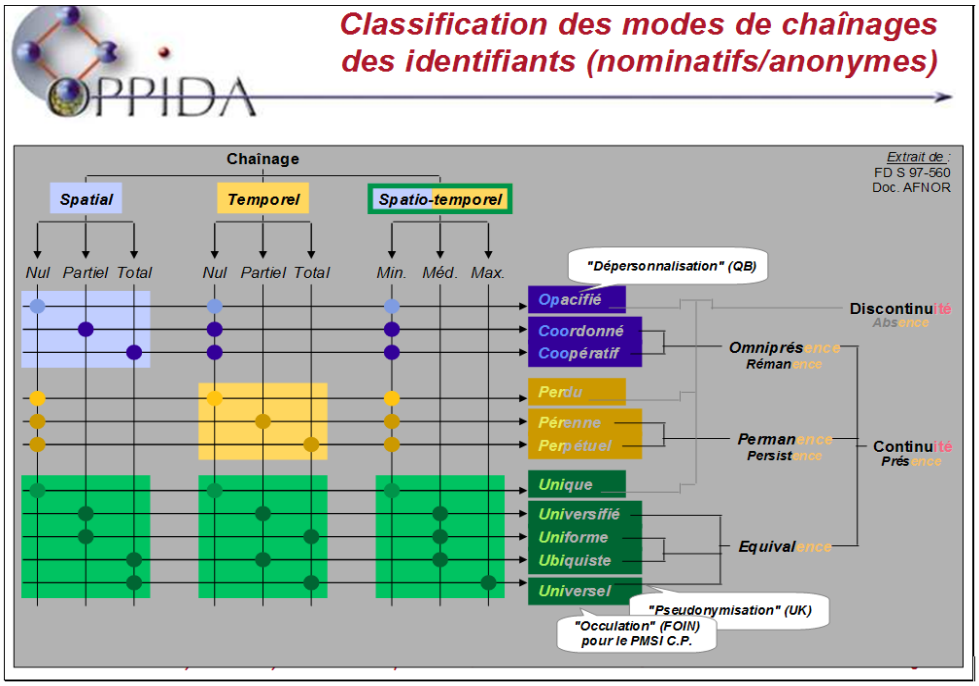


FIG. 2: Classification des modes de chaînage des identifiants (nominatifs ou anonymes) selon la nature du chaînage spatial et/ou temporel appliqué dans la sphère Santé-Social

En guise d'illustration, deux projets d'anonymisation dans la sphère Santé-Social résument assez bien tout le spectre des possibilités offertes par ces techniques d'anonymisation avec chaînage d'épisodes et donc constitution de trajectoires :

- **anonymisation de la base PMSI de l'Assurance Maladie :** à partir du recueil au fil de l'eau de l'ensemble des actes médicaux dispensés, il a été possible, en prenant les dispositions juridiques, organisationnelles et techniques nécessaires, de chaîner tous les épisodes de soins de tous les patients ayant été hospitalisés tant en hôpital public qu'en clinique privée : on pourrait donc résumer cette illustration comme étant un exemple de chaînage spatio-temporel

de type toujours et partout , pour la thématique soins et la discipline soins hospitaliers ; il est à noter que dans un tel cas de chaînage quasi-universel (toujours et partout), où les identifiants sont toujours et partout les mêmes, on peut parler de pseudonymisation plutôt que d'anonymisation, et aussi qu'un double-hachage est impératif et enfin qu'un sur-hachage est très recommandé chaque fois qu'une extraction de la base universelle est effectuée pour mener une nouvelle étude (par thématique ou bien par thématique et par discipline) ;

– **anonymisation pour un observatoire des RMIstes :** à partir de l'observation périodique des situations des personnes bénéficiant du RMI, il a été possible de comparer un extrait de la base constituant cet observatoire à un échantillon considéré statistiquement représentatif de la population des RMIstes, pour étudier statistiquement l'utilité de l'outil RMI ou, plus exactement, l'efficience du I de RMI : on pourrait donc résumer cette illustration comme étant un exemple de chaînage spatio-temporel de type jamais et nulle part (puisqu'il a été décidé de ne pas corréler les observations périodiques entre elles), ou sorte de randomisation des identités, pour la thématique prestation sociale et la discipline bénéficiaires du RMI ; il est à noter qu'un chaînage en mono-hachage a été nécessaire mais suffisant, à condition d'avoir à la fois la garantie techniquement irrévocable et juridiquement irréfragable, que les échantillons observés sont jetables et ne seront jamais chaînés entre eux ; sinon la robustesse de l'anonymisation ad hoc mise en place serait fortement remise en question.

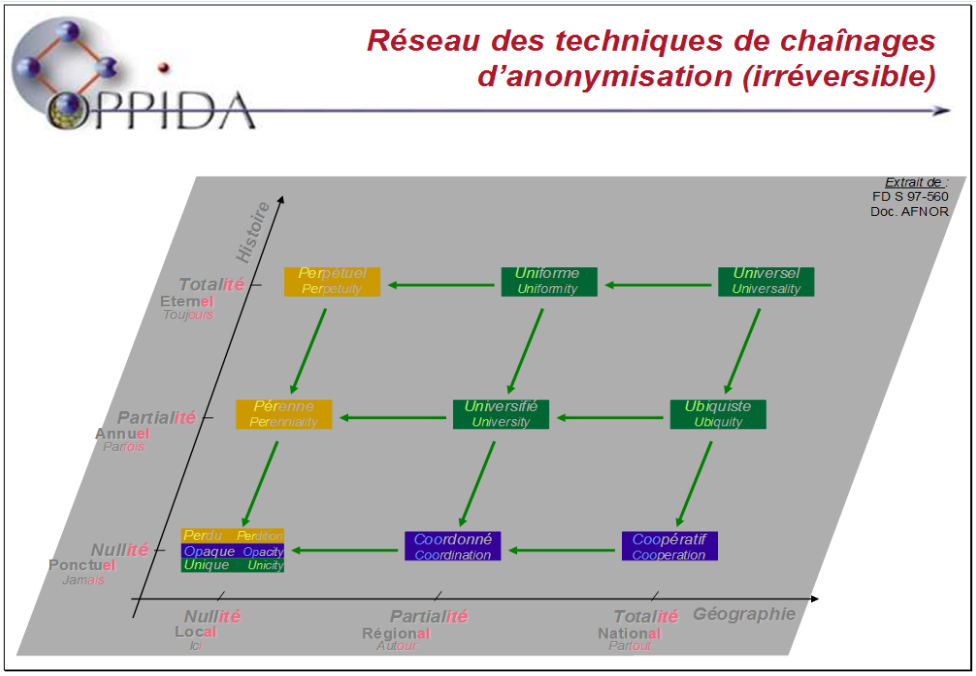


FIG. 3: Illustration du réseau des combinaisons possibles (en cascade) des techniques de chaînages d'anonymisation (irréversibles)

Il est par ailleurs parfaitement possible d'envisager des anonymisations en cascade ou, plus exactement, des hachages en cascades (ou sur-hachage) chaque fois que des exigences de robustesse l'imposent comme introduit précédemment.

Comme le montre la figure 3, il est possible de faire décroître le niveau d'application d'un chaînage en réappliquant un hachage selon une segmentation spatiale et/ou temporelle. On obtient tout un réseau de possibilités de techniques de chaînages en anonymisation (puisqu'il s'agit bien de hachage) permettant de dégrader successivement (par re-hachages successifs) la portée d'un chaînage et offrant ainsi une solution générique pour réduire la majorité des risques d'inférences décrits précédemment (déduction, induction et abduction).

Comme le montre la figure 4, de façon similaire, il est également possible de faire décroître le niveau d'application d'un chaînage en réappliquant non plus un hachage mais un chiffrement selon une segmentation spatiale et/ou temporelle similaire. On obtient tout un réseau de possibilités de techniques de chaînages en pseudo-anonymisation (puisqu'il s'agit de chiffrement) permettant de dégrader successivement (par re-chiffrements successifs) la portée d'un chaînage et donc de réduire les risques d'inférences décrits précédemment, mais tout en permettant de recouvrer la portée initiale du chaînage originel (par déchiffrement successifs) à condition d'avoir pris soin au préalable de conserver en lieu sûr le chemin cryptographique correspondant, c'est-à-dire : la suite ordonnée des clés successives nécessaires aux déchiffrements successifs.

6 Conclusions

En conclusion de cet article présenté à la conférence C&ESAR 2008, il est utile de résumer la portée des études et réflexions menées depuis plus d'une dizaine d'années, en matière d'anonymisation dédiée à la sphère Santé-Social que ce soit au niveau de l'Assurance Maladie (cf. [2,4,5,6]), des concepts et des normes (cf. [7,8]) ou des études épidémiologiques en milieu hospitalier (cf. [1,3,9,10,11]) :

- **pas de confidence (médicale) sans confiance (singulière) :** selon un adage du secteur, en général il n'est pas possible, pour un médecin, d'envisager recueillir la confidence (on dit la plainte) de son patient si ce dernier n'accorde pas une confiance singulière dans l'exploitation informatique qui en sera faite ;
- **distinction entre confidentialité-discrétion et confidentialité-séclusion :** il est alors essentiel de pouvoir faire la distinction entre la confidentialité partageable (ou partagée) en pleine confiance entre le patient et son médecin ou le corps médical (ce qui impose une certaine discrétion obtenue classiquement par du chiffrement-déchiffrement (réversible)) et la confidentialité impérative et absolue, nécessaire lorsque la confidence médicale est utilisée à d'autres fins que la seule dispensation des soins, telles que des études statistiques et/ou épidémiologiques : ceci impose une séclusion obtenue en empilant d'autant plus de fonctions à sens unique que le chaînage sera dangereusement universel ;
- **distinction entre pseudo-anonymisation (réversible) et véritable anonymisation vraie (irréversible) :** ces deux formes de confidentialité complémentaires, la confidentialité-discrétion et la confidentialité-séclusion, peuvent aboutir à des notions opposées de l'anonymisation : la première ne permettant qu'un simili-anonymat ou pseudo-anonymat est de peu d'intérêt dans les contextes et environnements considérés ici et pour la sensibilité hautement critique de la sphère Santé-Social ; la seconde peut, en garantissant les différentes facettes de

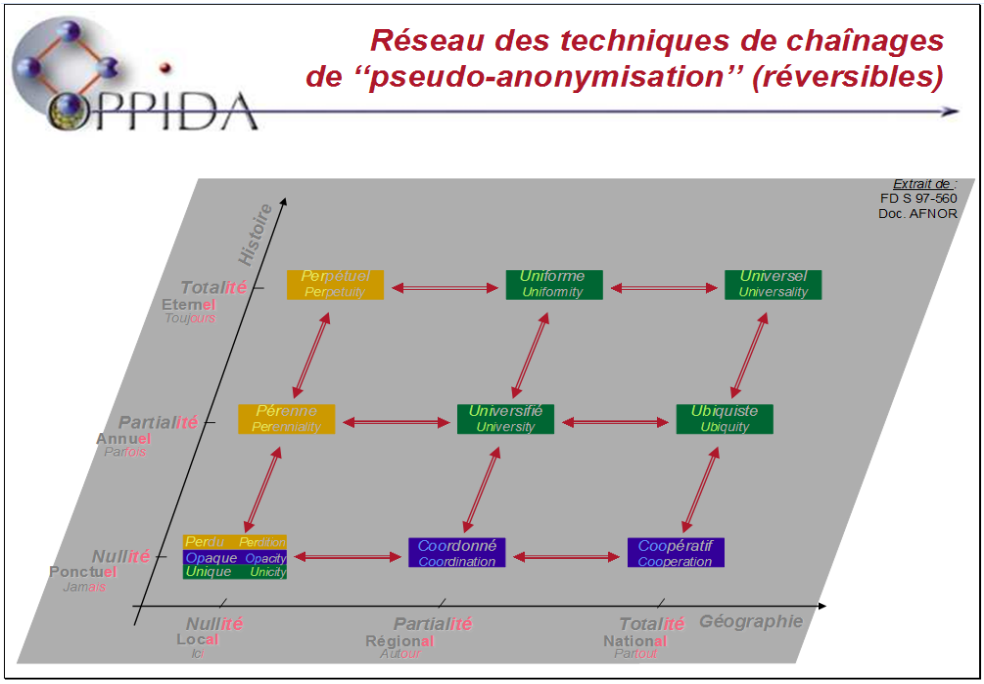


FIG. 4: Illustration du réseau des combinaisons possibles (en cascade) des techniques de chaînages de pseudo-anonymisation (réversibles)

sa robustesse, permettre une véritable anonymisation au vrai sens du terme, c'est-à-dire a priori irréversible, si ce n'est par des techniques très évoluées d'inférences ; mais dans ce cas des sur-hachage appropriés et adaptés à l'analyse de risques de désanonymisation permettront de parer la plupart des menaces véhiculées par les risques d'inférence intentionnellement malignes ;

– **distinction entre pure anonymisation au sens strict du terme (sans chaînage) et pseudonymisation (avec chaînage) :** cette seconde forme de confidentialité, la confidentialité-séclusion, peut aboutir à deux familles d'anonymisation : d'un côté, l'anonymisation qui manipule des anonymats stricts, c'est-à-dire lorsqu'il n'y a pas de chaînage entre les épisodes concernant un même individu et, de l'autre côté, la pseudonymisation qui, dans un périmètre espace-temps donné, manipulera toujours-et-partout le même pseudonyme pour identifier (de façon anonymisée) le même individu ; la pseudonymisation est la forme offrant la plus grande potentialité d'utilisation mais donc la plus grande variété de risques de violation de la propriété de confidentialité, que ce soit en violation de la confidentialité-discrétion en retrouvant, par exemple, une pathologie non saisie dans le système d'information (i.e., indiscrétion médicale), ou bien en violation de la confidentialité-séclusion en retrouvant, par exemple, l'identité d'un individu référencé toujours-et-partout de la même façon pourtant anonymisée vraie dans le système d'information (i.e., risque de désanonymisation).

Confidence versus confiance et confidentialité-discrétion versus confidentialité-séclusion, puis pseudo-anonymisation versus vraie anonymisation ou anonymisation versus pseudonymisation sont autant de composantes de la robustesse à examiner avec grande précaution dans ces contextes et environnements aussi sensibles de la sphère Santé-Social, avant de pouvoir espérer placer une confiance justifiée dans des traitements dits de confiance (cf. trusting trusted computing?) de données à caractère personnel de type médical, social, médico-social, socio-sanitaire ou encore économico-socio-sanitaire.

En dernière conclusion, il faut rappeler que si les données à caractère personnel, surtout celles ayant une sensibilité toute particulière comme les données de santé, peuvent être anonymisées de façon robuste appropriée (simple pseudo-anonymisation, sinon anonymisation vraie, voire pseudonymisation des identifiants) et donc avec une confiance justifiée appropriée; en revanche, elles ne doivent jamais être considérées comme parfaitement anonymes avec une confiance aveugle inappropriée, du fait que les utilisations qui pourraient en être faites pourraient risquer tôt ou tard de *laisser transpirer* (i.e., faire réapparaître) les identités réelles des individus concernées et dont elles devaient pourtant protéger l'intimité et garantir l'anonymat.

Remerciements

Remerciements à tous ceux qui, de près ou de loin, ont pu contribuer à cet édifice en construction et encore inachevé qu'est l'anonymisation dans la sphère Santé-Sociale et, notamment, aux relectures de la première version de cet article qui avait été soumise pour publication et présentation au cours des journées C&ESAR 2008.

Références

1. Gensbittel, M.H., Riandey, R, Quantin, C. : Appariements sécurisés : statisticiens ayez de l'audace. Courrier des statistiques, n° 121-122, mai-décembre 2007.
2. Ferragu, A., Mission d'exploration sur les procédures et modalités permettant d'assurer la reconstitution des séquences de soins et le chaînage des informations individualisées se rapportant à un même bénéficiaire. CNAMTS, rapport de mission, 20 février 1998.

   Ce rapport, dans la perspective de la mise en œuvre des dispositions de la convention d'objectifs et de gestion entre la CNAMTS et l'État, pour la période 1997-1999, relative à la modernisation du système d'information de l'Assurance Maladie, aborde le problème des séquences de soins, au travers du chaînage des informations de santé détenues par l'Assurance Maladie, en introduisant les notions d'anonymat et d'inférence, comme évoqués dans les précédentes propositions du CESSI-CNAMTS.
3. Quantin,C., Cohen, O. : Construction d'un identifiant pour le projet Inforare.

   Il s'agit d'un algorithme permettant de constituer un identifiant à partir d'informations nominatives à composantes familiales (cf. brevet n°04,09584 déposé le 9 septembre 2004 par C. Quantin, DIM Dijon, et O. Cohen, HC Forum) et rendues anonymes à partir du logiciel ANONYMAT développé par le DIM du CHU de Dijon et agréé en 1996 par la CNIL (Commission Nationale de l'Informatique et des Libertés) ainsi que par le SCSSI (Service de la Sécurité des Systèmes d'Information).
4. Trouessin, G., Tierces Parties de Confiance interopérables (TPCi) – interoperable Trusted Third Parties (iTTP). Colloque InfoSec'99 (Securicom), La Défense – Paris, 1-3 juin 1999.

   Cette publication aborde les définitions et concepts cryptographiques de bases nécessaires à la coopération minimales, ou interopérabilité, des différentes formes de Tierces Parties de Confiance, notamment

les TPC pour garantir la confidentialité dont une variante novatrice, les Tiers d'Anonymat, est consacrée au respect de la vie privée des individus, et des patients/malades, par le biais des techniques d'anonymisation.

5. Trouessin, G., Dependability Requirements and Security Architectures for the Healthcare/Medical Sector. SafeComp'99, Toulouse, 27-29 septembre 1999.

   Cette publication aborde les besoins actuels et orientations futures indispensables à la mise en place et au maintien global de la confiance dans les nouveaux SIS (Systèmes dInformations de Santé), allant de la Sécurité-Security des SIS (dérivée de la Sécurité Informatique) jusqu'à leur Sécurité-Safety (issue de la Sûreté de Fonctionnement). Un cas d'architecture sûre, inspiré du Data Matching Agency australien, y est abordé à travers la nécessité de pouvoir garantir l'anonymat des données de santé manipulées tout en offrant, éventuellement, la possibilité d'effectuer des croisements anonymisés.

6. Trouessin, G., Anonymisation & Risques d'inférence : concepts et terminologie & caractérisation et méthodologie. CNAMTS, document CESSI/GT/2000.02, avril 2000.

   Ce rapport, voulu dans la continuité des travaux autour de l'anonymisation menés depuis des années au CESSI/CNAMTS, tant sur plan théorique que sur le plan pratique avec le développement de la Fonction d'Occultation d'Informations Nominatives (FOIN), aborde les risques de désanonymisation sous l'angles des risques d'inférence en nominatif et/ou en confidentialité.

7. AFNOR – Association Française de NORmalisation, Anonymisation : glossaire et démarche d'analyse et d'expression de besoins. Fascicule de Documentation n° FD S97-560 paru en 2000 (issu de travaux effectué pour la CNAMTS).

   Le travail présenté dans ce document est issu d'une réflexion menée en deux temps au sein du groupe ad hoc d'Afnor Sécurité/Santé créé en 1997 et supervisé jusqu'en 1999, par la Commission Générale Afnor Informations de Santé et, depuis début 2000, rebaptisé Groupe d'Experts Sécurité des Systèmes d'Information de Santé supervisé par la Commission de Normalisation Afnor Informations de Santé. La réflexion menée en deux temps a consisté d'abord, de fin 1997 à mi-1998, à obtenir un consensus sur la terminologie établie en matière d'anonymisation pour établir un glossaire le plus exhaustif et flexible possible. Dans un second temps, en 1998-1999, les travaux ont abouti à une démarche d'analyse et d'aide à l'expression de besoin en matière d'anonymisation, à l'attention des différents acteurs de l'informatique de santé.

8. CEN – Comité Européen de Normalisation, A.U.R.T.A.F. Anonymity User Requirements for Trusted Anonymisation Facilities. CEN-Health Informatics / WG-Security, Safety and Quality (document CEN/TC251/WGiii N00-014).

   Proposition de sujet de travail européen rédigé par G.Trouessin, validé par le groupe GE-SSIS d'AFNOR et proposé par la France, puis accepté, le 14 mars 2000 par Bruxelles comme sujet de travail prioritaire.

9. Quantin, C., Allaert, F.A., Avillach, P., Fassa, M., Riandey, B., Trouessin, G., Cohen, O., Building Application-Related Patient Identifiers : What Solution for a European Country ? AMIA 2007 Annual Symposium, Biomedical and Health Informatics from Foundations to Applications to Policy, 10-14 novembre 2007, Chicago ; à paraître dans International Journal of Telemedicine and Application 2008, Article ID 678302, 1-5.

10. Quantin, C., Fassa, M., Coatrieux, G., Trouessin, G., Allaert, F.A., Combining hashing and enciphering algorithms for epidemiological analysis of gathered data. IMIA 2008, Methods of Information in Medicine.

11. Quantin, C., Fassa, M., Coatrieux, G., Riandey, B., Trouessin, G., Allaert, F.A., Chaînage de bases de données anonymisées pour les études épidémiologiques multicentriques nationales et internationales : proposition d'un algorithme cryptographique. À paraître in Revue d'Epidémiologie et de Santé Publique.

Interprétation du vla.4/van.5 dans le domaine du logiciel

Pascal Chour

Direction centrale de la sécurité des systèmes d'information
51 boulevard de La Tour-Maubourg
75700 Paris cedex SP

1 Objet de l'instruction

Cette instruction précise la position du schéma français vis-à-vis des composants d'assurance AVA_VLA.4 et AVA_VAN.5.

2 Références

CC v2.3 Critères Communs Parties 1-2-3 et CEM ; version 2.3 ; août 2005 ; réf. : CCMB-2005-08-001 à 004

CC v3.1 Critères Communs Parties 1-2-3 et CEM ; version 3.1 ; septembre 2007 ; réf. : CCMB-2007-09-001 à 004

3 Problématique

Si la communauté des évaluateurs, utilisateurs, développeurs et certificateurs a une bonne compréhension de la signification du AVA_VLA.4 / AVA_VAN.5 (par la suite, VLA4 et par extension, " niveau VLA4 ") dans le domaine de la carte à puce et des microcontrôleurs sécurisés, on constate qu'il n'en est pas de même pour le domaine logiciel à la date de rédaction de ce document. Pour tenter d'en comprendre les raisons, il faut rappeler quelques-unes des particularités des évaluations de cartes et les mettre en regard de ce qui constitue l'état actuel des évaluations logicielles :

1. Le nombre d'évaluations de cartes et composants dans le cadre des schémas français et allemand est très important et représente probablement la catégorie de produits pour laquelle il y a le plus fort retour d'expérience. Le nombre d'évaluations de logiciels se répartit sur de nombreuses catégories (pare-feux, systèmes d'exploitation, signature, chiffrement, etc.). Elles sont réalisées au sein de nombreux schémas d'évaluation ce qui fait qu'il est plus difficile de consolider un retour d'expérience.
2. Les évaluations de cartes et de composants se font essentiellement au niveau VLA4. Les évaluations logicielles sont le plus souvent faites à des niveaux AVA_VLA2 ou 3, sauf lorsqu'il existe des hypothèses simplificatrices (cf. 5).
3. Les évaluations de cartes et de composants se font par rapport à des profils de protection peu nombreux ce qui facilite les comparaisons. En revanche, pour des mêmes types de produits logiciels, il peut exister de nombreux profils de protection ce qui exclut toute possibilité de comparaison.

4. Les utilisateurs de cartes évaluées et certifiées sont très concernés par la façon dont se déroulent les évaluations (en particulier, par la compétence des centres d'évaluation) et par le résultat des analyses de vulnérabilités. Les évaluations de produits logiciels sont essentiellement commanditées dans le but de disposer d'un avantage concurrentiel. Il est rare que les utilisateurs finaux soient impliqués dans ce processus. Les développeurs (et les schémas de certification) n'ont donc pas d'intérêts particuliers à pousser les centres d'évaluation vers le haut en termes de compétence sur les attaques.
5. Lors des évaluations de cartes, il n'y a généralement pas d'hypothèses simplificatrices ayant pour conséquence l'élimination de certains chemins d'attaque. Dans le cas des évaluations de logiciels, il y a généralement de nombreuses hypothèses simplificatrices (hypothèses sur l'environnement technique et organisationnel) qui limitent la portée des attaques pouvant être réellement menées sur le produit. Autrement dit, la résistance d'un produit à des attaques de niveau VLA4 peut être systématiquement atteinte du fait qu'en pratique, les hypothèses éliminent les possibilités d'attaques.
6. La communauté impliquée dans les évaluations de cartes et de composants est très active dans la spécification des méthodes d'évaluation des cartes, y compris sur l'aspect lié aux attaques. En pratique, le travail de cette communauté permet de définir l'état de l'art des attaques de haut-niveau. Il est difficile d'identifier une telle communauté dans le domaine du logiciel.
7. La notion " d'erreur de construction ", pouvant entraîner des vulnérabilités, est mal acceptée dans le monde de la carte (même s'il en existe). Par la force des choses, les utilisateurs de logiciels en sont venus à considérer comme une fatalité l'existence d'erreurs de construction (bogues) entraînant la création de vulnérabilités qui représentent l'essentiel de celles qui sont découvertes. Un effet pervers de ce phénomène est que, même si les fonctions de sécurité d'un produit sont susceptibles de résister à un niveau élevé d'attaque, la confiance dans ce résultat est en pratique assez faible car des erreurs de construction non détectées durant l'évaluation sont susceptibles de remettre en cause l'efficacité de ces fonctions, alors même qu'elles ne sont pas forcément impliquées dans ces erreurs[1].

De ces constats, il apparaît que la signification du VLA4 dans le domaine logiciel peut être très variable (ce constat est d'ailleurs également valable pour les autres niveaux de AVA_VLA). Pratiquement, deux produits fonctionnellement identiques peuvent être certifiés avec le même niveau AVA_VLA alors que dans un cas, la sécurité du produit reposera sur des hypothèses d'environnements techniques et organisationnels très fortes tandis que ce ne sera pas le cas pour l'autre.

Cette souplesse autorisée par les CC (dans une certaine mesure, ils sont conçus pour) peut créer des situations que l'on pourrait qualifier de trompeuses pour l'utilisateur final, pas forcément au fait de ces subtilités. Le schéma français peut contribuer à éviter que ce type de situation se perpétue en proposant des règles pour l'acceptation des dossiers de demande de certification. Il faut néanmoins être conscient de la limite de ces règles, les accords de reconnaissance actuels (CCRA, SOG-IS) obligeant la DCSSI à formellement reconnaître les certificats des pays émetteurs qui ne les appliqueraient pas.

[1] Il peut sembler surprenant que l'évaluation ne soit pas en mesure d'identifier ces anomalies. Il faut rappeler que sur un logiciel de grande taille et très répandu, typiquement un système d'exploitation, il existe de nombreux experts qui passent un temps important à étudier et à travailler sur chacun de ces sous-systèmes. Leur puissance collective d'analyse est nettement supérieure à celle que peut mettre en œuvre un centre d'évaluation, faisant travailler un nombre limité de personnes en temps contraint pour trouver des vulnérabilités.

4 Illustrations

Considérons un équipement de chiffrement de réseau IP. Une architecture possible serait la suivante :

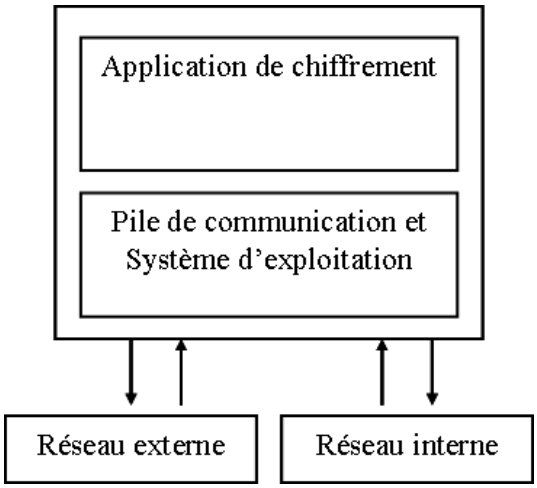


FIG. 1: Architecture 1.

Dans cette architecture, le processus de chiffrement partage une même pile protocolaire pour ses communications entrantes et sortantes, l'ensemble s'exécutant sur un système d'exploitation commun.

Dans l'architecture suivante, l'environnement des moyens de communication IP et de chiffrement s'exécute sur des plates-formes physiquement séparées, ces plates-formes étant reliées par un bus de communication matériel dédié.

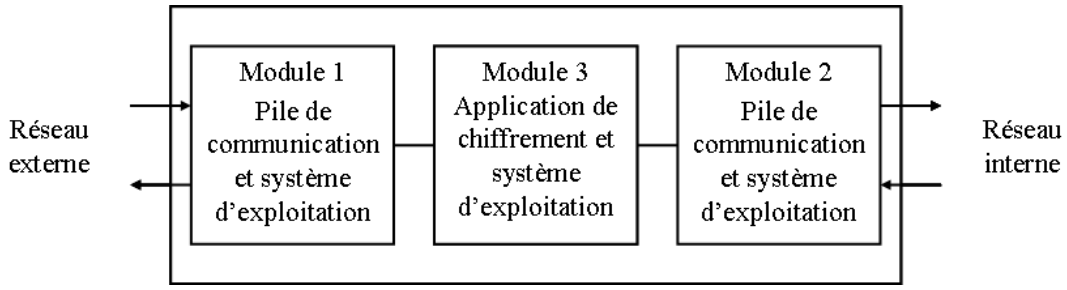


FIG. 2: Architecture 1.

Sans entrer dans le détail des hypothèses de conception, il est probable que, même à hypothèses identiques (hypothèse de confiance " a priori " sur le système d'exploitation et la pile de communication par exemple), l'architecture 2 inspirera une confiance plus grande que la première. Une raison est qu'il est intuitivement plus difficile de compromettre les modules 2 et 3 à partir d'une vulnérabilité exploitée sur le module 1 que dans le cas de l'architecture 1 où une vulnérabilité sur n'importe quelle composante (système d'exploitation, pile de communication, application) est susceptible de compromettre toutes les autres composantes. Cette confiance plus importante est déterminée par le concept de défense en profondeur, élément qu'il est difficile de prendre en compte dans une évaluation CC.

Cet exemple est une illustration des différences de signification possibles pour un composant AVA_VLA.x et particulièrement, VLA4. D'autres cas peuvent être évoqués :

- Un produit complexe inspire en général moins confiance qu'un produit simple. Une illustration peut être tirée du monde de la carte. Il existe des cartes dont les applications s'appuient sur un interpréteur Javacard et d'autres dont les applications sont programmées en natif. Un interpréteur Javacard propose des fonctionnalités à un niveau d'abstraction plus élevé que le processeur ce qui entraîne une plus grande complexité du code et des tests que pour une carte programmée en natif. Inversement, si l'on ne considère que le point de vue de l'application, sa programmation sera simplifiée si elle s'appuie sur une Javacard puisqu'elle utilise des primitives plus riches que celles proposées par le processeur. On voit par cet exemple que le critère de complexité doit être interprété au cas par cas.
- Beaucoup d'applications logicielles s'appuient sur d'autres produits pour s'exécuter et, en particulier, sur un système d'exploitation qui peut être généraliste (Microsoft® Windows®, Linux). En quantité de code exécutable, les applications en question peuvent ne représenter qu'une part infime du code effectivement présent (et exécuté) par le produit complet (application, système d'exploitation, matériel et logiciels embarqués). La pratique montrant que la probabilité d'avoir un bogue susceptible d'entrainer une vulnérabilité exploitable sur les produits sous-jacents est importante, le risque est grand que les fonctions de sécurité implantées dans les applications en question puissent devenir inopérantes.

5 Acceptation des demandes de certification VLA4

La problématique exposée dans les paragraphes précédents n'est pas nouvelle (elle est aussi ancienne que les critères d'évaluation). Pour autant, elle n'a pas forcément donné lieu à une doctrine bien établie concernant ce qu'il est raisonnable d'accepter en matière d'évaluation au niveau VLA4. La raison est que, jusqu'à présent, il n'a pas été exposé de règles applicables de façon systématique et objective. Ce document n'a pas pour objectif de fournir de telles règles mais, néanmoins, le schéma français a souhaité évoquer quelques principes directeurs sur ce qu'il semble raisonnable d'accepter ou de refuser au titre d'une demande de certification au niveau VLA4.

Principe 0 : la DCSSI se donne le droit de refuser une demande de certification visant le niveau VLA4 qu'elle estimerait trompeuse.
Principe 1 : l'architecture du produit doit contribuer à la sécurité en faisant en sorte que les fonctions de sécurité du produit ne puissent être mises en défaut que si au moins deux vulnérabilités distinctes sont exploitées. *Exemple : voir illustrations dans le présent document, partie 4.*
Principe 2 : les hypothèses sur l'environnement technique doivent être réalistes et en tout ou partie vérifiables.
Principe 3 : les hypothèses sur l'environnement technique (matériel et logiciel) doivent être détaillées. Le développeur doit fournir à l'évaluateur la preuve du réalisme de ces hypothèses. *Exemples : l'hypothèse " système d'exploitation de confiance " doit être détaillée. La question de la protection mémoire doit être développée : selon le système d'exploitation utilisé, comment le programme évalué gère-t-il le problème du swap, du coredump, d'un éventuel " debugger " qui serait en train de le tracer, des copies mémoire d'éléments secrets que certaines fonctions de librairies pourraient réaliser (que se passe-t-il si on passe une clé de chiffrement à une fonction C non écrite par le développeur ? Risque-t-elle d'être copiée en mémoire et de perdurer au-delà du moment où l'original de la clé est surchargé par des " 0 " au sein du programme principal ?). L'hypothèse "le matériel de confiance " doit également être détaillée. On pourra notamment proposer des hypothèses du type "le processeur est exploité dans son mode nominal de fonctionnement. Dans ce mode, le processeur gère au minimum deux niveaux de privilège (un niveau utilisateur et un niveau superviseur) " et " toute transition vers un mode du processeur autre que son mode nominal est impossible ". Dans le même ordre d'idée, il existe des publications montrant l'exploitation de vulnérabilités touchant en particulier l'implémentation de la cryptographie en exploitant les propriétés du cache mémoire, les mécanismes de prédiction de branchement, la rémanence des mémoires et les traitements associés au calcul des sous-clés.* *Voir également la note associée au principe 4.*
Principe 4 : les hypothèses d'environnement organisationnel doivent être réalistes pour le type de produit concerné. *Note : il est acceptable de définir plusieurs niveaux de résistance aux attaques selon les fonctions de sécurité proposées par le produit plutôt que de clamer que toutes les fonctions de sécurité visent le niveau VLA4 et introduire pour ce faire des hypothèses d'environnement très simplificatrices (du point de vue de l'analyse de vulnérabilité). La cible de sécurité sera alors multi-niveau pour AVA_ VLA.*
Principe 5 : un produit dont la résistance aux attaques de fonction de sécurité VLA4 ne repose que sur des hypothèses d'environnement n'est pas acceptable.
Principe 6 : une mesure (métrique à définir) de la complexité du logiciel évalué devrait être fournie et vérifiée par le laboratoire.
Principe 7 : l'existence de compétences couvrant les fonctionnalités de sécurité du produit chez le développeur devrait être validée.
Principe 8 : avant toute demande de certification au niveau VLA4, le commanditaire devrait valider avec la DCSSI l'acceptabilité de sa cible de sécurité.

Offline dictionary attack on TCG TPM weak authorisation data, and solution

Liqun Chen[1] et Mark Ryan[1,2]

[1] HP Labs, UK
[2] University of Birmingham

Abstract The Trusted Platform Module (TPM) is a hardware chip designed to enable PCs achieve greater security. Proof of possession of values known as authData is required by user processes in order to use TPM keys. We show that in certain circumstances dictionary attacks can be performed offline on authdata. In this way, an attacker can circumvent some crucial operations of the TPM, and impersonate the TPM owner to the TPM, or the TPM to its owner. For example, he can unbind data or migrate keys without possessing the required authorisation data, or fake the creation of TPM keys. This means that any application that relies on the TPM may be vulnerable to attack.

We propose a new solution and some modifications to the TPM specification to prevent the offline attacks, and we also provide the way to integrate these modifications into the TPM command architecture with minimal change. With our solution, the user can use a password-type of weak secret as their authData, and the TPM system will be still safe.

1 Introduction

The Trusted Platform Module (TPM) is a hardware chip specified by the Trusted Computing Group (TCG) industry consortium, with the aim to enable computers to achieve greater levels of security than was previously possible. There are 100 million TPMs currently in existence, mostly in high-end laptops. The TPM stores cryptographic keys and other sensitive data in its shielded non-volatile memory. Application software such as Microsoft's BitLocker and HP's HP ProtectTools use the TPM in order to guarantee security properties.

The Trusted Platform Module (TPM) stores cryptographic keys and other sensitive information in shielded locations. Processes running on the host laptop or on other computers can use those keys in controlled ways. To do so, such processes have to prove knowledge of the relevant authorisation data, called authData. The authData is chosen by the user process, and sent encrypted to the TPM. The TPM stores the authData along with the relevant keys or other sensitive information. In any communication between the user and TPM which requires owner authorisation, the authData is used as a HMAC key.

Although authData is 160 bits and is therefore capable of being a high-entropy value, it may be that actual authData is derived from human-memorable passwords, and is therefore low-entropy. The TPM specification [5] has no restrictions about the entropy expected in authData. If low-entropy data is used, an attacker could try successively to guess all the possible values of the authData, and verify each guess in turn. Such attacks are called dictionary attacks. The TPM specification stipulates that TPM manufacturers should implement resistance to dictionary attacks on authorisation data, for example, by permitting only a small number of incorrect guesses per minute.

We show that guesses of low-entropy data can be verified offline, so that the resistance offered by the TPM is ineffective. We propose a new solution and modifications to the TPM specification to prevent the offline dictionary attacks, and we also provide the way to integrate these modifications into the TPM command architecture with minimal change. With our solution, the user can use a password-type weak secret as their authData, and the TPM system will be still safe.

2 The offline dictionary attack

We show that an attacker that can observe some dataflow between the TPM and the user processes can perform an offline dictionary attack. Specifically, if the attacker can observe :

- A command containing proof of possession of authData (typically during an *Object-Independent Authorization Protocol* (OIAP) session), and the TPM response;
- Or, a command containing proof of possession of the shared secret associated with such authData (typically during an *Object-Specific Authorization Protocol* (OSAP) session), and the TPM response;

then the attacker can conduct an offline dictionary attack to discover the authData. This is possible because the attacker can confirm its guess of authData by reconstructing the authorisation HMACs based on the guessed authData, and comparing with the observed HMAC. If they are equal, that confirms the guess. The TCG specification assumes that the required traffic observations by the attacker are possible, and was intended to protect against them.

Moreover, the TPM specification uses shared secrets derived from authData to protect new authData supplied by the user for newly created keys. Once a single piece of authData is compromised, any new authData protected by it is also compromised. Therefore the technology specified in the current TPM specification version 1.2 can only work safely with the condition that users always choose a strong secret as their authData and never disclose their authData to any mistrusted entities. Obviously this condition restricts the TPM applications.

We propose a new solution and modifications to the TPM specification to prevent the offline dictionary attacks, and we also provide the way to integrate these modifications into the TPM command architecture with minimal change. With our solution, the user can use a password-type weak secret as their authData, and the TPM system will be still safe.

3 Password-based key agreement

Password-based authenticated key agreement is a cryptographic primitive that addresses this kind of problem. It is specified in IEEE P1363.2 [1] and ISO/IEC 11770-4 [2]. There are a large number of such key agreement protocols. The most attractive one for the current purpose is SPEKE (Simple Password Exponential Key Exchange) by Jablon [3,4]. It enables a pair of entities A and B to establish a strong shared secret s based on a weak secret w that they already share.

To do this, they run a Diffie-Hellman key exchange protocol, in which they use the shared weak password to compute a group generator. Let G be a finite field group of prime order q and prime modulus p, where q is at least 160 bits, and p is at least 1024 bits, such that $q \mid p-1$. Let H be a secure hash function $H : \{0,1\}^* \rightarrow G$. We assume that the values q, p and the function H are known to A and B (they are not secrets), and also that the weak shared secret w is known to A and B. The exchange proceeds as follows.

- A creates a new random $x \in Z_q^*$ and sends $H(w)^x \bmod p$ to B. For simplicity, we omit "mod p" in $H(a)^b \bmod p$ for any values a and b in the remaining part of the paper.
- A responds by creating a new random $y \in Z_q^*$ and sends $H(w)^y$ to B.
- Now, A and B can each compute the strong shared secret as $s = H(w)^{xy}$.

4 Solving the offline authData attack

Our proposed new method of TPM resistance to offline dictionary attack on weak authorisation data has been developed based on the SPEKE protocol. We assume that the user process can introduce new authData to the TPM in a reliable way (e.g., by encrypting it with an already established TPM key). It remains to demonstrate how the user process can later prove its knowledge of the authData when it wants to execute an authorised command.

In our basic solution, we directly make use of the SPEKE scheme between the TPM and user process to derive a strong secret based on the weak authData. SPEKE applied to this situation would look as follows. Every time a user process executes a command requiring authorisation with authData d, the user process and the TPM engage in the SPEKE protocol using d as the weak shared secret w. The user process chooses a random x and sends $H(d)^x$ to the TPM; the TPM chooses a random y and sends $H(d)^y$ to the user. Then they each compute the strong secret $s = H(d)^{xy}$ as explained above. After that, the value w is replaced by the value s as a HMAC key.

We have a number alternatives of the above basic solution, each of which achieve a unique requirement, which might be needed by different applications. We will give the details in the full paper.

Alternative 1 : Password-based key retrieval. The aim of this version is to reduce the TPM's computation task in the basic solution. Instead of choosing a fresh y each time, the TPM has a long-term secret key y, called the "authData key", which is used to process multiple authData values. At the time the user process sends the encrypted newly chosen authData d, the TPM stores d and returns the value $H(d,t)^y$ where t is some object-specific text (such as the name or the digest of public key of the object). Every time a user process executes a command requiring authorisation, it creates a new random x and sends $H(d,t)^x$ to the TPM, together with the command requiring authorisation using $H(d,t)^{xy}$ as the HMAC key.

The object-specific text t is required to avoid an *online* attack. Without t, an attacker could use the TPM as an oracle to confirm his guess d' of some authData d by introducing a new object with authData d', and comparing $H(d)^y$ with $H(d')^y$. The TPM should not resist such an attack.

Alternative 2. Password-based proof of knowledge. The aim of this solution is to avoid the TPM's long-term authData key being directly used by multiple users in multiple sessions, in order to enhance safety of the key. Again, the TPM has a long-term secret key y. At the time the user process sends the encrypted newly chosen authData d, the TPM stores the value $k = H_0(d,y,t)$ where H_0 is a secure hash-function $H_0 : \{0,1\}^* \rightarrow Z_q$ and t is the object-specific text; then the TPM replies with the message $H(d)^k$, and discards d. To demonstrate authorisation for a command, the user process chooses a random x and sends $H(d)^x$, and uses the value $H(d)^{kx}$ as the HMAC key for the command requiring authorisation. The user process may use the same x across several authorisations, or it may pick a new x each time.

Alternative 3. Password-based proof of knowledge without long term key. The aim of this solution is to further reduce the TPM's computation and storage task in the previous solutions. At the time the user process sends the encrypted newly chosen authData d, the TPM chooses a random value $k \in Z_q^*$ and stores it. The TPM replies with the message $H(d)^k \in G$, and discards d. To demonstrate authorisation for a command, the user process chooses a random x and sends $H(d)^x \in G$, and uses the value $H(d)^{kx} \in G$ as the HMAC key for the command requiring authorisation. The user process may use the same x across several authorisations, or it may pick a new x each time.

5 Integration with TPM command architecture

We show how to integrate our solutions into the TPM command architecture, requiring minimal changes to the existing command set. We illustrate that for "Alternative 3". The changes listed are described from the point of view of the TPM :

1. Commands that introduce newly created authData require to be changed. The incoming and outgoing operands and their sizes do not need to be changed, but the TPM should not store the authData d. In the place of d it stores the new random k that it created, and it discards d.
2. Commands that require proof of possession of authData also require to be changed. The value $H(d)^x$ computed by the user process should be supplied as an additional incoming operand. The TPM then retrieves the value k that it stored in place of the authData, and it uses $H(d)^{kx}$ in the HMAC key in order to reconstruct and verify the incoming HMAC.

6 Conclusion

We have proposed a new solution for TCG TPM resistance to offline dictionary attack on weak authrisation data. The new solution offers the following advantages over the existing solutions : (1) The new solution protects the weak authorisation data from offline dictionary attacks. (2) The new solution can be integrated into the TPM command architecture, requiring minimal changes to the existing command set.

Acknowledgments. Many thanks to Carsten Rudolph for pointing out the necessity of the object-specific text t in alternatives 1 and 2.

Références

1. IEEE P1363.2/D26 Draft Standard for Specifications for Password-based Public Key Cryptographic Techniques. `grouper.ieee.org/groups/1363/passwdPK/index.html`
2. ISO/IEC 11770-4 :2006. Information technology – Security techniques – Key management – Part 4 : Mechanisms based on weak secrets.
3. David Jablon. Strong password-only authenticated key exchange. *Computer Communication Review* **26**(5) :5–26. ACM SIGCOMM. October 1996.
4. David Jablon. Extended password key exchange protocols immune to dictionary attack. In *Proceedings of the Sixth Workshops on Enabling Technologies : Infrastructure for Collaborative Enterprises (WET-ICE '97)*, pages 248–255. IEEE Computer Society, 1997.
5. Trusted Computing Group. `www.trustedcomputinggroup.org`

Un panorama des architectures informatiques sécurisées et de confiance

Guillaume Duc[1,2] et Ronan Keryell[1]

[1] Institut Télécom – Télécom Bretagne
CS 83818 – 29238 Brest Cedex 3, France
[2] Orange Labs
42 rue des Coutures – 14000 Caen, France
guillaume.duc,ronan.keryell@telecom-bretagne.eu

Résumé Du fait de l'ubiquité actuelle des ordinateurs, la confiance dans leur bon fonctionnement devient un problème essentiel. Longtemps assurée principalement par logiciel, la sécurité commence à être déployée plus profondément au niveau matériel.

D'un côté, de plus en plus d'ordinateurs sont aujourd'hui équipés d'une puce TPM dont l'objectif est de permettre l'informatique de confiance. D'un autre côté, ces dernières années, de nombreuses architectures sécurisées, permettant de garantir l'intégrité et/ou la confidentialité de processus contre des attaques matérielles ou logicielles, ont été présentées aussi bien dans le monde académique que dans le monde industriel.

Après une présentation des principales architectures introduites dans le cadre de l'informatique de confiance ainsi que les principales architectures sécurisées, nous montrerons quelles sont les relations entre ces deux approches et donnerons quelques exemples d'applications.

Mots-clés : informatique de confiance, architectures sécurisées

1 Introduction

Avec la diffusion de plus en plus importante des ordinateurs dans de nombreux aspects de la vie et de la société, les besoins de faire confiance au bon fonctionnement de ces outils sont de plus en plus importants. L'informatique de confiance devient un sujet d'actualité.

Cette confiance peut servir à ou être exigée par différentes entités. Tout d'abord, elle peut être désirée par l'utilisateur qui veut, par exemple, que son ordinateur n'exécute que des programmes qu'il a approuvés ou soit véritablement inutilisable en cas de vol afin de protéger ses données personnelles.

La banque de l'utilisateur peut également avoir intérêt à disposer d'un mécanisme qui lui permette d'avoir confiance dans le bon fonctionnement de l'ordinateur de son client afin de s'assurer que les opérations effectuées en ligne proviennent bien de l'utilisateur et pas d'un programme malicieux par exemple.

Un site de jeu en ligne va également avoir besoin d'un moyen de faire confiance au fait que la partie cliente du jeu qui s'exécute chez le joueur n'est pas modifiée par ce dernier de manière à lui procurer un avantage dans la partie.

Le calcul distribué (de type grille) vise à mutualiser des moyens de calcul inaccessibles autrement à des utilisateurs isolés. En exécutant à distance des morceaux de son application, un utilisateur obtiendra plus rapidement ses résultats ou pourra faire des simulations plus importantes. Mais qu'est-ce qui prouve qu'un administrateur distant n'espionnera pas ou ne modifiera pas les résultats des calculs ? À l'opposé, si un utilisateur offre de la puissance de

calcul, qu'est-ce qui prouve que les applications qui tournent sur son ordinateur sont bien de celles qu'il autorise ?

Enfin, plus controversé, un distributeur de musique ou de film en ligne va avoir besoin de faire confiance au fait que le logiciel de lecture sur l'ordinateur du client respecte bien les licences d'utilisation des oeuvres numériques achetées ou louées, et que le système d'exploitation 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 [5,7,8], Dallas DS5000 [11], ibm 4758 [36,23], TrustNo 1 [29], Xom [32,33,47], Aegis [39,38], CryptoPage [28,31,13,15,12]...). Bien que leur objectif premier soit 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

Introduction D'après son site Web [41], 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.

Le tcg est divisé en plusieurs groupes de travail visant à sécuriser différents domaines touchant à l'informatique : postes de travail, serveurs, 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 fabriquants de microprocesseurs (Intel, amd), de mobiles (Nokia...), de cartes à puce (Gemalto...) ou des opérateurs téléphoniques (France Télécom, Vodaphone...).

Architecture proposée

Fonctionnalités fondamentales D'après le TCG [45], 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*) qui sont à la base de la confiance dans une plate-forme. 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.

Le Trusted Platform Module Pour les ordinateurs, l'architecture de confiance proposée par le TCG s'appuie sur un circuit baptisé *Trusted Platform Module* (TPM [46]) qui 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. Au démarrage du TPM, ces registres sont mis à zéro. La commande `TPM_Extend` (fournie par le TPM) permet de les mettre à jour de façon sûre. 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 (c'est-à-dire l'objet mesuré et la valeur correspondante). 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 [2].

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 [34] 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 (PCR) 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 à 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 majeure des spécifications du TPM apporte un certain nombre de nouveautés [44] :
- *Direct Anonymous Attestation* : ce mécanisme, décrit dans [9], 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. Il permet donc d'améliorer la protection de la vie privée de l'utilisateur en réduisant la possibilité d'associer les différentes attestation effectuées par son TPM.
- 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 (pour aider à contrer certaines attaques par rejeu par exemple).
- Stockage et restauration de contexte : permet à plusieurs applications d'utiliser les fonctionnalités du TPM en même temps.

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* [43] permettent à des opérateurs réseaux d'imposer des politiques d'accès basées sur l'intégrité (mesurée à l'aide d'un TPM) des terminaux souhaitant se connecter, 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 [42] 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 fabriquant pour ce qui concerne la partie logiciel et le 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.

Critiques Les critiques contre les propositions du TCG ont été très nombreuses et virulentes et le couple *TCPA/Palladium* [1] est devenu une bête noire dans l'informatique et est souvent associé aux systèmes de gestion des droits sur les œuvres numériques (DRM).

Par exemple, dans [37], STALLMAN écrit un essai particulièrement virulent contre le concept d'informatique de confiance, préférant même utiliser le terme d'informatique traître (*treacherous computing*). Il cite notamment le fait que des programmes, locaux ou distants, peuvent contrôler les programmes ou les documents auxquels l'utilisateur aura accès et la façon dont il y aura accès. L'utilisateur n'aura plus le contrôle sur les programmes qu'il souhaitera exécuter, ou alors (s'il choisi de désactiver son TPM), ne pourra plus accéder à de nombreux services nécessitant l'attestation de l'état de sa plate-forme, voire même au réseau (avec les travaux sur le *Trusted Network Connect*).

Dans [17], FELTEN montre que les fonctionnalités offertes peuvent être préjudiciables à l'écosystème informatique. Par exemple, le stockage scellé peut être utilisé pour empêcher l'interopérabilité entre des logiciels de différents fabriquants. 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.

La définition des configurations de plate-forme considérées comme dignes de confiance pose également problème. Outre le fait que l'établissement de ces listes (en effet, chaque programme ou service qui reçoit les mesures peut décider lesquelles sont dignes de confiances) pourrait faire l'objet de pression de certains fabriquants de matériels ou vendeurs de logiciels, leur mode de fonctionnement est plus adapté aux logiciels propriétaires qu'aux logiciels libres. En effet, il existe souvent très peu de variantes d'un exécutable propriétaire et donc il est facile d'établir la liste des mesures (résumés cryptographiques) de ces dernières. Or, dans le cas d'un logiciel libre, même s'il est déclaré comme digne de confiance, certains modes de diffusion de celui-ci, et notamment les distributions recompilant à l'installation le logiciel à partir de ses sources (mécanisme de *ports* dans les systèmes BSD, meta-distributions GNU/Linux comme par exemple *Gentoo*), font qu'il est difficile, voire impossible, d'établir la liste de toutes les mesures possibles correspondantes à une même version du code source du logiciel.

2.2 Intel Trusted Execution Technology

La *Trusted Execution Technology* (TXT), initialement connue sous le nom de *LaGrande*, présentée par *Intel* dans [25] définit des améliorations au niveau plate-forme afin de fournir les bases pour construire des plates-formes de confiance permettant d'aider à protéger des informations sensibles contre des attaques logicielles. Les spécifications sont disponibles dans [26].

Les objectifs L'objectif (d'après [25]) 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. Il est précisé que pour les périphériques USB, cette protection est possible en chiffrant les informations envoyées par le clavier ou la souris à l'aide d'une clé de chiffrement partagée entre le périphérique d'entrée et le gestionnaire d'entrée d'un domaine protégé.

Graphiques protégés (*Protected Graphics*) qui permettent de protéger les données envoyées par une application protégée vers l'écran contre une application espionne et de permettre à l'utilisateur de s'assurer que ce qu'il voit provient bien de la bonne application.

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

Lancement protégé (*Protected Launch*) qui permet de contrôler le lancement des parties critiques du système d'exploitation et des logiciels dans l'environnement d'exécution protégé.

Le matériel La *Trusted Execution Technology* est basée sur un certain nombre de modifications matérielles de la plate-forme.

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 des accès directs à la mémoire depuis un périphérique (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.

Fonctionnement D'après [26], les améliorations proposées dans la plate-forme permettent :
- le lancement de *Measured Launched Environments* (MLE),
- la protection de ces MLE contre la corruption

Ces améliorations se basent notamment sur l'utilisation d'une extension du jeu d'instruction du processeur, *Safer Mode Extensions* (SMX), et qui intègrent les fonctions suivantes :
- lancement mesuré d'un MLE,
- protection et stockage sécurisé de ces mesures,
- contrôle par le MLE des tentatives de modifications de lui-même.

Pour permettre la mesure des MLE, la plate-forme n'utilise pas la racine de confiance pour la mesure (RTM) définie par le TCG, mais une autre, dynamique, afin de faciliter la prise de mesures, car les MLE peuvent être lancés à n'importe quel moment de la vie de la plate-forme, indépendamment des logiciels qui ont été lancés précédemment.

Lors du lancement d'un MLE, la plate-forme charge deux modules en mémoire : le MLE en question et un code d'authentification (*Authentication Code*). Ce code d'authentification n'est utilisé que lors de la mesure et de la vérification et est signé par le vendeur du chipset. Le code d'authentification est chargé au sein d'une mémoire résistante aux attaques, située à l'intérieur même du processeur qui vérifie sa signature. Ce code vérifie ensuite que la configuration du processeur et du chipset est acceptable puis mesure et lance le MLE. Il vérifie également, grâce à une série de règles (*Launch Control Policy*), stockées dans la mémoire non volatile du TPM, qu'il a le droit de charger le MLE demandé dans l'état actuel de la plate-forme. Les mesures effectuées lors du lancement d'un MLE sont stockées au sein du TPM présent sur la plate-forme. Une fois le MLE lancé, il est protégé contre un accès direct (DMA) non autorisé d'un périphérique à sa mémoire grâce à la technologie VT-d (*Intel Virtualization Technology for Directed I/O*).

Utilisation En 2007, Intel a fourni un outil nommé tboot [27] montrant comment utiliser la technologie TXT pour effectuer le lancement mesuré et vérifié d'un noyau ou d'un gestionnaire de machine virtuel, en l'occurrence l'hyperviseur XEN.

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.

IBM 4758 L'IBM 4758 [36,23,35,24] 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, des accélérateurs cryptographiques, 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.

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, mémoires 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.

Best BEST, dans [5,6,7,8], pose les bases d'un microprocesseur cryptographique disposant d'un bus de données chiffré permettant d'exécuter un programme chiffré stocké en mémoire. L'objectif en terme de sécurité est d'assurer la confidentialité du code et des données du programme en question contre des attaques matérielles (l'attaquant pouvant manipuler tous les périphériques extérieurs au processeur).

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.

Un mécanisme optionnel de destruction de la clé est ajouté au processeur afin de rendre inutilisable le programme s'il a été exécuté plus d'un certain nombre de fois ou au bout d'un certain temps. De même, des instructions de destruction de la clé sont insérées dans le jeu d'instruction du processeur afin qu'un attaquant qui essaierait de modifier le programme chiffré ait une forte probabilité de les déclencher et donc de rendre le processeur inutilisable.

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.

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 [11], 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.

Kuhn, dans [30] propose une attaque pratique contre le DS5002FP. Le principe de cette attaque est de présenter au microcontrôleur de nombreuses instructions chiffrées et, en analysant les réactions de celui-ci, d'identifier certaines des instructions en clair correspondantes (une remise à zéro du microcontrôleur est effectuée entre chaque essai afin de repartir d'un état connu). Une fois certaines instructions intéressantes identifiées, on peut essayer de construire de petits programmes chiffrés ayant pour objectif de faire fuir plus d'informations et au final, déchiffrer complètement la mémoire sur un port de sortie. Cette attaque est possible car le chiffrement s'effectue octet par octet et donc il est relativement rapide de parcourir l'ensemble des chiffrés possibles et de ne modifier qu'une seule instruction à la fois.

ARM TrustZone TrustZone [3,4,22] 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 coeur 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 coeur 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.

TrustNo 1 Kuhn [29] 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. Tout d'abord, durant l'exécution d'une instruction lue depuis un segment mémoire chiffré, si une interruption survient, le processeur sauvegarde son état interne (y compris les valeurs des registres et du compteur de programme) dans une pile interne et efface les données sensibles avant de rendre la main au système d'exploitation. Au retour de l'interruption, le processeur restaure son état interne depuis cette pile et peut ainsi reprendre l'exécution du processus interrompu, sans que le système d'exploitation n'ait pu avoir accès aux données du processus chiffré.

Pour permettre le changement de contexte d'un processus vers un autre, le processeur fournit deux instructions, la première qui permet de chiffrer puis de sauvegarder en mémoire l'état du processeur stocké en haut de la pile interne (c'est donc l'état du dernier processus chiffré interrompu), et la seconde permettant d'effectuer l'opération inverse (chargement d'un état chiffré depuis la mémoire, déchiffrement et stockage dans la pile d'état interne). Ainsi le système d'exploitation peut changer de processus mais ne peut pas avoir accès, grâce au chiffrement, au contenu des états internes sauvegardés. Ces états internes sont chiffrés avec une clé aléatoire stockée dans une table des processus gérée au sein du processeur.

Une autre instruction permet au système d'exploitation de dupliquer l'état d'un processus et ainsi permettre la création d'un nouveau processus (opération similaire à l'appel système *fork* d'Unix). Enfin une autre instruction permet à un processus sécurisé de réaliser un appel système (et donc une interruption) sans effacer l'état interne. Ainsi, le processus chiffré peut passer les paramètres de l'appel système au système d'exploitation via des registres. Il a néanmoins la responsabilité d'effacer les autres registres ne servant pas à cet appel afin de ne pas laisser fuir d'informations.

Cependant, cette architecture ne traite pas non plus des problèmes liés à la protection de l'intégrité des données stockées en mémoire.

XOM Lie, Thekkath, Mitchell et Lincoln [32] 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 [3].

Dans [33], Lie, Thekkath et Horowitz présentent un système d'exploitation, xomos, qui n'a pas besoin d'être de confiance et qui permet d'exploiter les fonctionnalités offertes par Xom. Les auteurs ont implémenté leurs propositions en prenant comme base le système d'exploitation Irix (en ajoutant approximativement 1 900 lignes de code au noyau) et l'ont exécuté dans le simulateur SimOS. D'après leurs résultats, l'impact de Xom sur les performances des applications n'est que de 5 % en moyenne.

[3] Une 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.

Aegis Suh, Clarke, Gassend, Dijk et Devadas introduisent l'architecture Aegis [39] 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 vérification 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 [10]. L'utilisation d'arbres de hachage pour empêcher les attaques par rejeu a également été proposée par Lauradoux et Keryell dans l'architecture CryptoPage-2 [31].

Aegis dispose également d'une fonction d'attestation permettant à un programme s'exécutant dans le mode sécurisé de demander au processeur de signer un résultat produit par celui-ci. Cette opération permet ainsi d'attester que le programme, identifié par son résumé cryptographique, s'est bien exécuté sur un processeur Aegis (identifié par sa signature) et a produit le résultat en question en s'exécutant dans un mode protégé.

D'autres améliorations à l'architecture Aegis sont proposées dans [40]. Un mode d'exécution sécurisée suspendue (*Suspended Secure Processing*, ssp) est introduit pour permettre de suspendre temporairement les fonctions de sécurité (intégrité et confidentialité) afin d'exécuter plus rapidement certaines portions de code ne nécessitant pas de sécurité particulière.

Les auteurs présentent également une alternative aux clés asymétriques pour permettre au processeur de s'authentifier et de communiquer de façon sécurisée avec l'extérieur : les fonctions aléatoires physiques (*Physical Random Function*, puf). Ces fonctions, présentées dans [21,20,19], sont basées sur des mesures de variation de délais de propagation dans des portes logiques. Elles peuvent être réalisées de telle façon que leurs résultats soient propres à chaque processeur, compte tenu des différences dues aux procédés de fabrication. De plus, elles ne sont pas clonables. Les auteurs présentent comment utiliser ces puf afin d'identifier un processeur en particulier et comment permettre le chiffrement d'un exécutable pour un processeur particulier.

Enfin les auteurs ont implémenté Aegis sur un fpga, afin d'obtenir des mesures de performance et de vérifier l'espace nécessaire, en terme de portes logiques, pour implémenter les fonctions de sécurité. L'implémentation des différents mécanismes d'Aegis a nécessité un peu plus de 300 000 portes logiques.

HIDE Dans [48], 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 [18], 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.

CryptoPage L'architecture CryptoPage, initialement introduite par Keryell [28], vise également à protéger la confidentialité et l'intégrité de processus en présence d'un attaquant qui contrôlerait tous les composants extérieurs au processeur. Lauradoux et Keryell améliorent l'architecture afin de répondre aux attaques par rejeu contre la mémoire [31] en utilisant, comme pour Aegis, un mécanisme d'arbre de hachage.

Ensuite Duc et Keryell présentent une nouvelle version de l'architecture CryptoPage [12,14] 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).

Dans [12], des fonctionnalités supplémentaires sont présentées comme par exemple un mécanisme d'identification de programme permettant de s'assurer, malgré le chiffrement, qu'un programme sécurisé correspond bien à un programme donné, un mécanisme permettant de gérer les signaux logiciels, et un mécanisme permettant d'offrir aux applications un espace de stockage sécurisé contre le rejeu.

4 Lien entre architectures sécurisées et informatique de confiance

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

4.1 Architectures de confiance

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, et 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.

4.2 Architecture sécurisées

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 (c'est notamment le cas pour les architectures Aegis et CryptoPage) 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, si ce mécanisme fait défaut et si la confidentialité des applications est garantie (le code d'une application est chiffré avant et pendant son exécution), une application peut emporter elle-même un mécanisme de signature de ses résultats et le simple fait que les résultats aient pu être produits et signés prouve que l'application s'est bien exécutée sur une architecture sécurisée (qui a été capable de déchiffrer et d'exécuter le code), et si celle-ci garantie également l'intégrité de l'exécution des applications, que les résultats sont intègres. Ce mécanisme permet donc, si le destinataire du résultat fait confiance au fabriquant 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.) et la présence d'attaques logicielles ou matérielles. Dans la majorité des architectures sécurisées présentées, le système d'exploitation n'a pas besoin d'être digne de confiance.

Les architectures sécurisées permettent également une isolation forte entre les applications sécurisées. Une application, quelle soit sécurisée ou pas, ne peut pas espionner les données d'une autre application sécurisée, même si le système d'exploitation le permet, grâce au compartimentage et au chiffrement des données des applications sécurisées.

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 permettant une 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 [16].

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'informations personnelles à des personnes non autorisées.

Enfin, on pourrait concevoir des logiciels de gestion de droits numériques (DRM) libres dont les sources seraient publiés 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 intentionnée, 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. Ross Anderson. Trusted computing frequently asked questions, août 2003. `http://www.cl.cam.ac.uk/~rja14/tcpa-faq.html`.
2. 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, mai 1997.
3. ARM. Trustzone : Integrated hardware and software security. White Paper, juillet 2004. `http://www.arm.com/pdfs/TZWhitepaper.pdf`.
4. ARM. Trustzone technology overview, mars 2007. `http://www.arm.com/products/esd/trustzone_home.html`.
5. Robert M. Best. Microprocessor for executing enciphered programs. Technical Report US4168396, United States Patent, septembre 1979.
6. Robert M. Best. Preventing software piracy with crypto-microprocessors. In IEEE *Spring* CompCon *'80*, pages 466–469. IEEE Computer Society, février 1980.
7. Robert M. Best. Crypto microprocessor for executing enciphered programs. Technical Report US4278837, United States Patent, juillet 1981.
8. Robert M. Best. Crypto microprocessor that executes enciphered programs. Technical Report US4465901, United States Patent, août 1984.
9. 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, octobre 2004.
10. 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.

11. Dallas Semiconductor. *DS5002FP Secure Microprocessor Chip*, juillet 2006. `http://datasheets.maxim-ic.com/en/ds/DS5002FP.pdf`.
12. 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`.
13. 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, avril 2005.
14. 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, décembre 2006. `https://info.enstb.org/projets/cryptopage/documents/2006/ACSAC`.
15. 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)*, octobre 2006.
16. 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.
17. Edward W. Felten. Understanding trusted computing : Will its benefits outweigh its drawbacks ? IEEE *Security and Privacy*, 1(3) :60–62, mai 2003.
18. 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, septembre 2006.
19. Blaise Gassend, Dwaine Clarke, Daihyun Lim, Marten van Dijk, and Srini Devadas. Identification and authentication of integrated circuits. *Concurrency and Computation : Practice and Experience*, 16(11) :1077–1098, 2004.
20. Blaise Gassend, Dwaine Clarke, Marten van Dijk, and Srinivas Devadas. Controlled physical random functions. In *Proceedings of the 18th Annual Computer Security Applications Conference (ACSAC'02)*, pages 149–160. IEEE Computer Society, décembre 2002.
21. Blaise Gassend, Dwaine Clarke, Marten van Dijk, and Srinivas Devadas. Silicon physical random functions. In *Proceedings of the 9th ACM conference on Computer and Communications Security (CCS'02)*, pages 148–160. ACM Press, novembre 2002.
22. Tom R. Halfhill. ARM dons armor : Trustzone security extensions strengthen ARMv6 architecture. *Microprocessor Report*, 8/25/03-01, août 2004.
23. IBM PCI cryptographic coprocessor. `http://www-03.ibm.com/security/cryptocards/pcicc/overview.shtml`, mai 2007.
24. IBM 4764 PCI-X cryptographic coprocessor. `http://www-03.ibm.com/security/cryptocards/pcixcc/overview.shtml`, mai 2008.
25. Intel. *Intel® Trusted Execution Technology, Architectural Overview*, 2003. `http://www.intel.com/technology/security/downloads/arch-overview.pdf`.
26. Intel. *Intel® Trusted Execution Technology, Software Development Guide*, juin 2008. `http://download.intel.com/technology/security/downloads/315168.pdf`.
27. Intel. Trusted boot, octobre 2008. `http://sourceforge.net/projects/tboot`.
28. Ronan Keryell. CryptoPage-1 : vers la fin du piratage informatique ? In *Symposium d'Architecture (SympA'6)*, pages 35–44, Besançon, juin 2000.

29. M. Kuhn. The TrustNo1 cryptoprocessor concept. Technical Report CS555, Purdue University, avril 1997.

30. Markus G. Kuhn. Cipher instruction search attack on the bus-encryption security microcontroller DS5002FP. In ieee *Transaction on Computers*, volume 47, pages 1153–1157. ieee Computer Society, octobre 1998.

31. Cédric Lauradoux and Ronan Keryell. CryptoPage-2 : un processeur sécurisé contre le rejeu. In *Symposium en Architecture et Adéquation Algorithme Architecture (*SympAAA*'2003)*, pages 314–321, La Colle sur Loup, France, octobre 2003.

32. 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, octobre 2000.

33. 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, octobre 2003.

34. 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, août 2004.

35. Sean W. Smith. *Trusted Computing Platforms : Design and Applications*. Springer, 2004.

36. Sean W. Smith and Steve Weingart. Building a high-performance, programmable secure coprocessor. *Computer Networks*, 31(9) :831–860, avril 1999.

37. Richard Stallman. Can you trust your computer?, novembre 2007. `http://www.gnu.org/philosophy/can-you-trust.html`.

38. 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*, décembre 2003.

39. 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, juin 2003.

40. G. Edward Suh, Charles W. O'Donnell, Ishan Sachdev, and Srinivas Devadas. Design and implementation of the Aegis single-chip secure processor using physical random functions. In *Proceedings of the 32nd Annual International Symposium on Computer Architecture (*isca*'05)*, pages 25–36. ieee Computer Society, juin 2005.

41. Trusted Computing Group. `http://www.trustedcomputinggroup.org`, février 2007.

42. Trusted Computing Group : Mobile, mai 2008. `https://www.trustedcomputinggroup.org/groups/mobile/`.

43. Trusted Computing Group : Trusted Network Connect, mai 2008. `https://www.trustedcomputinggroup.org/groups/network/`.

44. Trusted Computing Group. tpm v1.2 specification changes, octobre 2003. `https://www.trustedcomputinggroup.org/groups/tpm/TPM_1_2_Changes_final.pdf`.

45. Trusted Computing Group. tcg specification architecture overview, avril 2004. `https://www.trustedcomputinggroup.org/specs/IWG/TCG_1_0_Architecture_Overview.pdf`.

46. Trusted Computing Group. Trusted platform module (TPM) main specification, part 1 : Design principles, part 2 : TPM structures, part 3 : TPM commands, mars 2006. `https://www.trustedcomputinggroup.org/specs/TPM/`.

47. Jun Yang, Lan Gao, and Youtao Zhang. Improving memory encryption performance in secure processors. IEEE *Transactions on Computers*, 54(5) :630–640, mai 2005.

48. 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, octobre 2004.

Apports de la virtualisation pour une plateforme informatique de confiance

Sébastien Gay

CELAR

Résumé La virtualisation, au delà de son intérêt en termes de mutualisation des machines et de flexibilité, parait également intéressante d'un point de vue sécurité pour améliorer le niveau de confiance d'un système. La propriété de cloisonnement qu'elle apporte, plus proche du matériel que les mécanismes de cloisonnement déjà présents dans les systèmes d'exploitation, commence à être utilisée pour isoler des mécanismes critiques ou même plusieurs instances d'OS de niveau de sécurité différents sur une même machine. Cet article va donc présenter tout d'abord les objectifs de la virtualisation et son fonctionnement, essentiellement sur plateforme x86. Puis, nous aborderons les vulnérabilités et limitations identifiées, avec un intérêt tout particulier pour la problématique de cloisonnement. Nous ferons également un point sur la problématique de la sécurité matérielle, qui a un fort impact sur la sécurité des solutions de virtualisation, ainsi que sur les nouveaux types de malwares utilisant la virtualisation à des fins malicieuses. Enfin, nous verrons quelques utilisations de la virtualisation dans des architectures de sécurité.

1 Introduction

La réalisation d'une plateforme informatique de confiance est une tâche difficile se heurtant actuellement au problème de la complexité des systèmes d'exploitations et des plateformes matérielles sous jacentes. Cette complexité croit de plus continuellement au fil des améliorations technologiques. Ainsi par exemple dans le cas d'une application critique, et quel que soit le soin apporté à son développement, le code s'exécutera toujours au sein d'un système d'exploitation et sur une plateforme matérielle le plus souvent standard. Le niveau de sécurité obtenu est ainsi au final celui de l'ensemble.

Il paraît cependant illusoire de vouloir s'assurer d'un niveau de sécurité élevé sur des systèmes d'exploitation de plusieurs millions de lignes de codes. Les bogues et vulnérabilités divers mis en évidence sur les principaux OS du marché illustrent bien cet état de fait, sans compter le risque de piégeage intentionnel que l'on ne peut écarter. Enfin, considérant que la sécurité de ces OS repose sur les mécanismes de sécurité matériels sous jacents, il parait encore plus difficile de vouloir s'assurer d'un niveau de sécurité élevé sur une plateforme matérielle également très complexe, aux composants d'origines diverses et aux fonctionnalités sans cesse en évolution. Ces composants matériels sont le plus souvent développés avec un souci de sécurité moindre, et on relève des problèmes de cohérence entre leurs différents mécanismes de sécurité. Comme nous le verrons par la suite, des travaux récents ont ainsi mis en évidence de nombreux problèmes au niveau du matériel permettant de contourner des mécanismes de sécurité logiciels par l'utilisation détournée de fonctionnalités matérielles. Dans ce contexte, la virtualisation apparait comme un moyen permettant d'augmenter le niveau de confiance d'une plateforme informatique en apportant :

- Une fonctionnalité de cloisonnement proche du matériel et implémentée en dehors de l'OS, et donc plus facilement évaluable que les mécanismes de cloisonnement interprocessus usuels internes à l'OS.
- Une possibilité de se placer en coupure des appels aux périphériques effectués par l'OS et donc d'assurer un certain niveau de filtrage vis à vis de ces appels, afin de limiter les risques quant à l'utilisation illicite de fonctionnalités matérielles.

L'objectif de cet article est donc de présenter de manière globale les apports de la virtualisation pour l'amélioration du niveau de sécurité d'une plateforme informatique, Nous aborderons tout d'abord les principes de la virtualisation et une description des principaux mécanismes de virtualisation possibles sur l'architecture x86, Puis nous ferrons un point sur les principales vulnérabilités et limites actuellement identifiées sur ces mécanismes. Enfin nous décrirons les utilisations qui peuvent en être faites dans le cadre d'architectures de postes informatiques d'un niveau de confiance élevé.

2 Qu'est ce que la virtualisation ?

2.1 Objectif de la virtualisation

Les technologies de virtualisation connaissent actuellement un développement important. Initialement introduites au milieu des années 60 par IBM sur ses mainframes, ces technologies se démocratisent depuis le début des années 2000 tant au niveau des clients que des serveurs avec des produits comme VMware, XEN ou plus récemment Microsoft Hyper-V.

L'objectif de la virtualisation est avant tout d'apporter un niveau d'abstraction entre des ressources matérielles et des systèmes d'exploitation les utilisant. Il devient ainsi possible aussi bien de faire fonctionner plusieurs OS sur la même machine physique que d'agréger des ressources physiques situées sur des machines distinctes pour faire fonctionner un OS unique [1]. Nous nous limiterons dans cet article à l'étude des virtualiseurs permettant le fonctionnement simultané de plusieurs OS sur une même machine physique, les ressources matérielles à partager étant ainsi entre autres le processeur, la mémoire, la carte réseau, les périphériques de stockage et les périphériques utilisateurs (écran, clavier et souris).

2.2 Positionnement de la virtualisation par rapport aux autres technologies similaires

Il est au premier abord difficile de catégoriser les différents produits permettant de faire fonctionner une ou plusieurs instances d'OS sur un même poste physique. La virtualisation n'étant pas la seule famille de produits apportant cette fonctionnalité, il est nécessaire de faire un tour d'horizon des différentes méthodes disponibles afin de comprendre pourquoi la virtualisation reste la plus intéressante dans notre cas de figure.

On identifie ainsi tout d'abord des mécanismes d'isolation (aussi appelés virtualisation au niveau du système d'exploitation) visant à isoler de manière plus ou moins poussée des contextes d'exécution au sein d'un même OS. Ces technologies sont la plupart du temps caractérisées par la définition de "cages" définissant des contextes d'exécution distincts au dessus d'un noyau unique. Les technologies usuelles rentrant dans cette catégorie sont les Chroot, V/Server ou Open VZ sous Linux ou encore les jail BSD. L'utilisation de ces techniques est courante

[1] D'autres types de virtualisation existent, que se soit au niveau du réseau, des applications ou du stockage. Nous nous concentrerons cependant ici sur la virtualisation de systèmes d'exploitation.

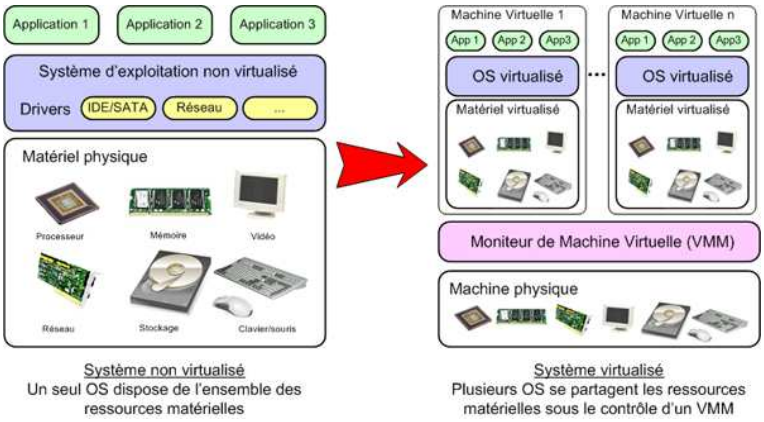


FIG. 1: Principes de la virtualisation

dans le monde des serveurs, pour isoler des services potentiellement vulnérables et ralentir une éventuelle compromission de tout le serveur en cas de problème (isolation du serveur Web, LDAP ou DNS par exemple). L'isolation apportée par ces technologies peut être plus ou moins élaborée, certaines technologies récentes apportant un "rendu" fonctionnel proche de la véritable virtualisation. Les mécanismes étant cependant répartis dans le noyau de l'OS cloisonné, il reste difficile d'avoir un bon niveau d'assurance quand au niveau d'étanchéité ainsi obtenu.

On appellera émulation de plateforme matérielle les technologies visant à simuler l'intégralité d'une architecture sur une autre architecture. Ce type de logiciels fournit une abstraction complète de plateforme informatique (processeur, chipset, périphériques, BIOS, etc.) aux applications émulées. Il est ainsi possible d'exécuter une application prévue pour une architecture informatique donnée sur une autre architecture, en contrepartie d'une forte perte de performances (par exemple une application prévue pour un pc/intel exécutée sur un mac/powerPC). Des exemples de produits de ce type sont QEmu[2] ou encore BOCHS. Le niveau de cloisonnement obtenu par de tels produits peut être au moins équivalent à celui des logiciels de virtualisation, mais la perte de performances reste problématique.

On trouvera également des mécanismes d'émulation d'API permettant par exemple d'émuler les API d'un OS sur un OS différent. L'idée est ici de rediriger les appels systèmes initiaux vers les appels systèmes de l'OS hôte afin de permettre l'exécution d'un logiciel sur un OS différent, avec cependant une compatibilité qui n'est pas complète. Des logiciels comme Wine ou Cygwin en sont des exemples. Ces solutions sont peu adaptées à des besoins de sécurité car trop intégrées à l'OS hôte.

Enfin, la virtualisation se caractérise par le fait de proposer aux systèmes virtualisés une architecture matérielle virtuelle complète, et de plus identique ou très proche du véritable système physique. L'idée est donc ici avant tout de partager des ressources matérielles entre différentes instances d'OS s'exécutant de manière concurrente, tout en laissant un maximum de code s'exécuter de manière native pour des raisons de performances. Contrairement aux émulateurs complets vus plus haut, on émule donc ici que ce qui est strictement nécessaire

[2] Uniquement dans le cas où QEmu fonctionne sans son module noyau kQEmu. Avec ce module, QEmu se comporte comme un virtualiseur en exécutant le code situé en espace utilisateur directement sur le processeur.

à assurer le partage et cloisonnement des ressources, l'essentiel du code pouvant s'exécuter directement. La contrainte qui en découle reste que les différents OS doivent être dédiés à la même architecture matérielle physique et qu'il n'est pas possible d'utiliser un OS prévu pour une architecture différente. Les produits les plus courants implémentant cette technologie sont la gamme de produits VmWare (Workstation, Player, ESX Server, etc.), la gamme de produits Microsoft (Virtual PC, Microsoft Virtual Server, Hyper- V), et le logiciel Open Source XEN et sa variante commerciale XenSource commercialisée par Citrix.

2.3 Types de virtualisation

Dans le cadre de la virtualisation, l'abstraction est mise en place par un composant logiciel appelé moniteur de machine virtuelle (Virtual Machine Monitor ou VMM) ou hyperviseur. Plusieurs instances d'OS s'exécutant en parallèle sur le même matériel, l'hyperviseur a donc en charge le contrôle d'accès, le partage ou parfois la fourniture de ces ressources par émulation, ce qui implique un contrôle complet de la plateforme matérielle (processeur, mémoire, entrées sorties etc.). On appellera OS ou instances invitées les OS virtualisés par l'hyperviseur.

Comme indiqué dans l'article de Popek et Goldberg[1], un hyperviseur se doit au final de respecter les 3 principes ou propriétés suivantes :

- **Équivalence** : tout programme exécuté dans l'hyperviseur doit avoir un comportement équivalent de celui qu'il a lorsqu'il est exécuté directement sur le matériel, excepté pour les aspects temporels. La virtualisation doit être transparente ;
- **Performance** : une écrasante majorité des instructions de l'OS invité doivent être exécutées directement sur la machine physique sans intervention de l'hyperviseur ;
- **Cloisonnement** : l'hyperviseur doit gérer l'ensemble des accès au matériel.

Suivant les ressources matérielles considérées, l'hyperviseur doit réaliser différents types de partage afin de créer l'illusion de virtualisation. Cela peut être un partage temporel, c'est-à-dire que l'accès à la ressource est successivement donné aux différentes instances invitées (cas du processeur par exemple). Cela peut être un partage statique définissant un partitionnement physique de la ressource considérée (par exemple partage de la mémoire ou du disque dur entre les instances). Enfin l'hyperviseur peut se placer en intermédiaire entre des périphériques virtuels et le périphérique physique en effectuant une action de partage spécifique (par exemple attribution des périphériques utilisateurs à une instance dite "courante" en fonction d'une combinaison de touches, gestion de files d'attentes pour le partage d'une carte réseau, etc.).

On distingue finalement deux types d'hyperviseurs, les différences d'architecture entre ces deux types ayant un impact direct sur leur niveau de sécurité :

- **Type I** : les hyperviseurs de type I s'exécutent directement au dessus du matériel, lancés immédiatement après la phase de boot. Après avoir pris le contrôle de la plateforme, ils démarrent les instances d'OS qui sont toutes virtualisées. Le plus souvent, une instance particulière (appelée domaine 0 ou domaine racine) disposera de droits particuliers lui permettant de piloter le mécanisme de virtualisation et d'intégrer les véritables drivers matériels. Cette instance de contrôle est donc extrêmement sensible d'un point de vue sécurité. Ce type d'architecture offre des performances plus importantes et un niveau de sécurité en théorie supérieur. On retrouve couramment ces hyperviseurs sur les serveurs.
- **Type II** : les hyperviseurs de type II s'exécutent au sein d'un OS standard appelé "hôte". Ces hyperviseurs ne sont donc pas lancés immédiatement après le boot mais en tant qu'application du domaine hôte. Ils vont donc s'appuyer sur les services de cet OS hôte pour piloter les accès au matériel et assurer le partage des ressources. Intégrés dans l'OS hôte, leur niveau de sécurité est au final difficilement quantifiable.

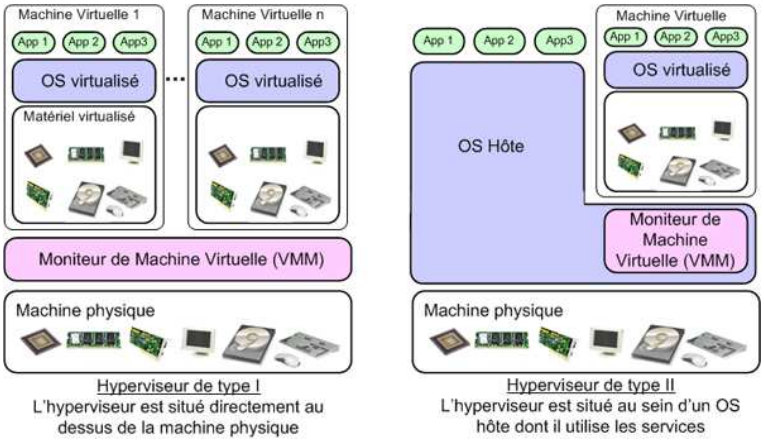


FIG. 2: Types de mécanismes de virtualisation

3 Fonctionnement de la virtualisation

3.1 Principes du "Trap & emulate"

La méthode traditionnelle de virtualisation se nomme "trap & emulate". Sur la plupart des architectures dites "classiquement virtualisables", il est possible d'interrompre l'exécution d'un programme en cas d'utilisation d'une instruction visant à modifier l'état du processeur. Il peut s'agir par exemple d'une instruction particulière identifiée comme privilégiée ou d'un accès à une zone mémoire sensible (tables de traduction mémoire, périphérique mappé, etc.). Pour ce faire, il suffit le plus souvent d'exécuter le code dans un mode du processeur non privilégié qui interrompra l'exécution dès qu'une telle instruction voudra être exécutée et donnera la main à l'hyperviseur.

Ainsi par exemple le noyau d'un OS, qui fonctionne habituellement en mode privilégié afin de pouvoir changer d'état le processeur (modification des mappings mémoire, de la segmentation, des bitmaps d'I/O, etc.) va désormais être exécuté en mode non privilégié. Chaque fois que ce noyau voudra modifier l'état du processeur, il va être interrompu et la main va être passée l'hyperviseur. Charge à cet hyperviseur d'émuler l'exécution de l'instruction privilégiée de manière sécurisée pour effectuer la partition des ressources et assurer le cloisonnement avant de redonner la main à l'OS.

3.2 Application à l'architecture x86 et limites

Le x86 dispose de 4 niveaux d'exécution appelés rings (ring 0 à ring 3), le ring 0 étant le ring d'exécution des instructions privilégiées. Le noyau des OS invités sera ainsi couramment déplacé en ring 1 (x86-32 : modèle "0/1/3") ou même ring 3 (x86-64 : modèle "0/3/3") afin que les instructions privilégiées provoquent des traps. Malheureusement, le x86 souffre d'un problème de séparation stricte entre les opérations privilégiées et non privilégiées. Ainsi 17 instructions privilégiées, si elles sont utilisées en dehors du ring 0, ne déclenchent pas de trap et empêchent donc l'hyperviseur de les émuler[2]. Leur utilisation au sein du noyau des OS invités provoque donc des résultats incohérents pouvant causer des erreurs ou des crashs système. D'importantes pertes de performances sont de plus à déplorer dues aux nombreux changements de contextes entre les rings lors de l'utilisation d'instructions privilégiées.

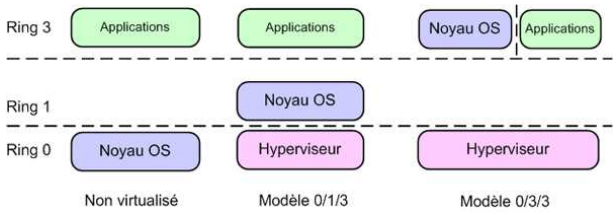


FIG. 3: Modèles de compression de rings

3.3 Méthodes de virtualisation purement logicielles sur x86

Afin de virtualiser l'architecture x86 malgré ces limitations, deux principales techniques sont couramment utilisées.

Traduction binaire dynamique. Le principe de cette méthode est d'effectuer une analyse et modification à la volée du code binaire des OS invités en détectant les instructions problématiques et en les remplaçant par des instructions simulant un comportement correct. Le code binaire source est traité par petits blocs et à la demande, pour n'avoir à traduire que le code réellement nécessaire et améliorer les performances. De plus seul le code noyau est à priori concerné car intégrant l'essentiel des instructions problématiques, le code en espace utilisateur pouvant être exécuté directement. Cette solution présente l'énorme avantage de pouvoir faire fonctionner à priori tout type de système d'exploitation sans adaptation, mais reste relativement complexe à implémenter et provoque de plus une baisse de performances due au nombre de changements de contexte. Les éditeurs intègrent cependant de nombreuses astuces visant à réduire au maximum ces pertes de performances. Enfin il demeure certains cas de figure difficiles à traiter (codes automodifiants par exemple).

Paravirtualisation. La paravirtualisation part du principe que l'OS invité va coopérer avec l'hyperviseur pour réaliser la fonction de virtualisation. L'OS va ainsi être modifié à l'avance pour faciliter la virtualisation. Donc au lieu d'analyser à la volée le code pour remplacer les instructions problématiques, ici on va directement modifier l'OS pour qu'il appelle les services de l'hyperviseur en lieu et place des véritables mécanismes du processeur. L'intérêt consiste en une amélioration des performances et une simplification certaine du mécanisme de virtualisation, mais par contre il faut bien sûr que l'OS soit modifié en conséquence (l'ampleur de ces modifications, cantonnée aux couches basses du noyau, reste quand même limité).
Concernant la prise en charge de ces modifications, deux options sont possibles :

- Modifier directement un OS afin qu'il utilise l'API de l'hyperviseur, et donc disposer de son code source. Cette voie est choisie par exemple par XEN avec la modification principalement de noyaux linux (xenokernels).
- Se mettre d'accord sur un standard d'API de paravirtualisation implémenté dans le noyau des OS directement par les développeurs de l'OS et non par les développeurs de l'hyperviseur. Les éditeurs de solutions de paravirtualisation se retrouvent ainsi déchargés de la tâche de modification de l'OS invité. Cette voie est maintenant adoptée par XEN et VMware via la définition et l'intégration d'une API standardisée (paravirt-ops) directement dans le noyau Linux.

Microsoft ayant opté pour la paravirtualisation dans son récent hyperviseur Hyper-V, et maitrisant bien sûr le développement de son OS, n'a pas rencontré cette problématique et a choisi de développer sa propre API[3].

3.4 Support matériel de la virtualisation

La virtualisation devenant relativement répandue sur x86, les fondeurs ont eu la volonté de pallier aux limitations de cette architecture en la rendant complètement virtualisable. De nouvelles fonctionnalités d'assistance matérielle à la virtualisation ont ainsi été implémentées rendant le x86 complètement virtualisable. Ces fonctionnalités sont intégrées à la plupart des processeurs récents, que se soit chez Intel (Intel VT-x) ou AMD (AMD-V). L'idée est ici double :

- Pallier aux limitations de l'architecture x86 en termes de virtualisation, en ajoutant un mode d'opération du processeur "VMX Root" spécifique à l'hyperviseur. L'OS virtualisé peut ainsi utiliser de nouveau les 4 rings d'exécution mais en mode "VMX non-root" où son exécution sera surveillée.
- Fournir des mécanismes complémentaires visant soit à améliorer les performances des virtualiseurs (par exemple gestion des mappings mémoire assistée) soit à apporter des mécanismes de sécurité supplémentaires qui ne sont pas réalisables en virtualisation logicielle (ajout d'une fonctionnalité d'IOMMU pour contrôler les accès mémoire des périphériques comme décrit plus loin).

La virtualisation matérielle tend ainsi à simplifier le développement d'un hyperviseur et donc à rendre plus facile son évaluation. La contrepartie est que l'on se repose encore un peu plus sur des mécanismes de sécurité matériels non maîtrisés. À noter que des travaux visant également à intégrer des fonctionnalités de virtualisation directement au niveau du périphérique et de son contrôleur sont en cours pour améliorer les performances et faciliter le développement de la fonction de partage du périphérique. L'impact en termes de sécurité sera cependant à examiner de près avant utilisation.

4 Vulnérabilités et limitations identifiées

Force est de constater que la sécurité n'a historiquement pas été la préoccupation majeure des développeurs des principales solutions de virtualisation. Conçus avant tout dans le but d'optimiser les ressources côté serveur, et à des fins de test et de flexibilité côté client, les aspects sécurité commencent tout juste à être traités sérieusement. Nous allons faire ici un point sur les menaces et vulnérabilités mises en évidence actuellement sur ces produits.

4.1 Menaces standards

Il est tout d'abord nécessaire de souligner qu'un système d'exploitation virtualisé est soumis au moins aux mêmes risques qu'un système d'exploitation installé sur une machine physique. Ainsi, toutes les menaces courantes s'appliquent : virus et troyens, vulnérabilités logicielles, menaces réseau (attaques sur les protocoles réseau, sur les applications, écoute des informations), vol du matériel, etc. Les OS virtuels sont néanmoins souvent moins bien sécurisés que leurs équivalent physiques. Le but n'étant pas ici de faire le point sur l'ensemble des menaces s'appliquant à un OS standard, on retiendra uniquement qu'il est nécessaire de sécuriser un OS virtualisé de la même manière que son équivalent physique (paramétrage correct, filtrage réseau, antivirus, etc.). Le cas particulier de l'OS Hôte ou privilégié sera détaillé par la suite.

4.2 Détection des environnements virtualisés

Historiquement, la détection de la virtualisation pour un code s'exécutant au sein de l'OS virtualisé a été la première problématique de sécurité mise en exergue sur ces produits. En

effet, la virtualisation est depuis longtemps utilisée dans l'analyse des malwares. Ceci a conduit certains malwares à inclure des fonctionnalités de détection de mécanismes de virtualisation, à l'image de ce qui se faisait déjà en termes de fonctionnalités anti debugger. Le malware arrête ainsi son exécution en cas de détection d'un virtualiseur, ce qui permet de retarder l'analyse et augmente sa durée potentielle de nocivité.

De nombreuses techniques permettant à un code de détecter le fait qu'il s'exécute sur un OS virtualisé ont ainsi été rendues publiques. Pour ce qui est des produits utilisant la paravirtualisation, la modification du code des OS invités rend triviale cette détection. Pour ce qui est des produits ne nécessitant pas de modification de l'OS invité, les techniques reposent essentiellement sur l'analyse de différences de comportement de la plateforme matérielle virtualisée par rapport à la plateforme physique. Les résultats d'exécution d'opérations spécifiques peuvent ainsi trahir la présence d'un hyperviseur, ainsi que l'analyse des temps d'exécution de certaines instructions qui diffèrent[4].

Au final, vu qu'il n'y a pas à priori de fuite d'information, l'impact en termes de sécurité est quand même plus limité que pour la mise en défaut du cloisonnement que nous verrons par la suite. Il a été montré qu'il s'avère impossible de rendre totalement invisible une fonctionnalité de virtualisation, mais la probable utilisation systématique de la virtualisation au sein des systèmes d'exploitation risque de rendre secondaire ce problème. La distinction entre plusieurs mécanismes de virtualisation imbriqués pour déterminer s'ils sont légitimes ou non est en effet beaucoup plus complexe que la simple détection du fait d'être virtualisé.

4.3 Mise en défaut du cloisonnement

D'un point de vue sécurité, la propriété la plus intéressante des produits de virtualisation reste leur capacité de cloisonnement. L'idée est ici de se protéger d'une instance virtualisée corrompue, celle ci ne pouvant attaquer une autre instance ou le système de virtualisation lui même. L'impact d'une faille dans le cloisonnement peut avoir des conséquences limitées à un simple déni de service ou plus grave en cas d'établissement de canaux de communication cachés ou de compromission d'une autre instance ou du mécanisme de virtualisation lui même.

Impact du type de virtualisation. Comme nous l'avons déjà vu, les hyperviseurs de type 1 s'exécutant directement sur le matériel sont en principe préférables aux hyperviseurs de type II. En effet, une architecture basée sur un hyperviseur de taille relativement réduite s'exécutant directement sur le matériel est plus simple à développer et évaluer, ces hyperviseurs intégrant ainsi directement les mécanismes de partage des ressources et de gestion du cloisonnement. Les produits de virtualisation de type II s'appuient eux sur les services de l'OS hôte (gestion mémoire, entrées sorties, ordonnancement, etc.). Comme pour les mécanismes de cloisonnement, la surface de code à considérer (potentiellement l'OS hôte complet) rend leur évaluation problématique.

Mécanismes de communication natifs. La plupart des produits implémentent nativement des mécanismes de communication entre les OS hôte et invités, parfois entre les invités eux mêmes. Ces mécanismes sont implémentés afin de permettre certaines fonctionnalités entre instances (accélération graphique, copier/coller, répertoires partagés, etc.). Ces canaux de communication, souvent non documentés sur les hyperviseurs du commerce, sont potentiellement sources de problèmes de sécurité en cas de problème de conception, d'implémentation ou tout simplement d'utilisation détournée de leurs fonctionnalités.

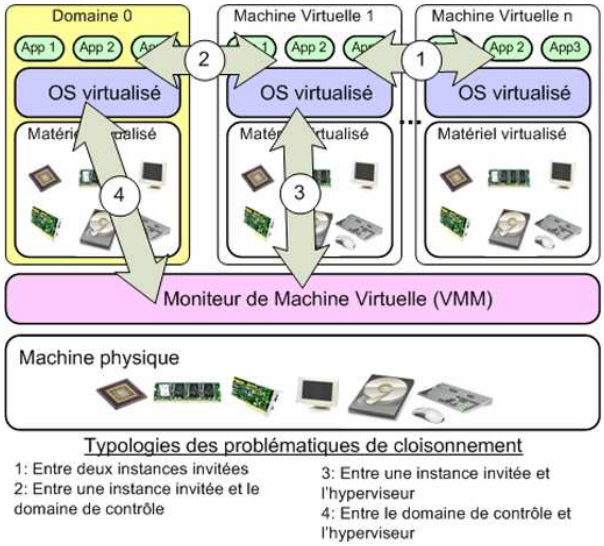


FIG. 4: Typologie des problèmes de cloisonnement

Par exemple les produits VMware implémentent un mécanisme de communication utilisé par les VMwares Tools. Le mécanisme, basé sur un appel I/O spécifique, a été analysé et décrit en grande partie, et permet d'établir des canaux de communication entre l'hôte et les instances virtualisées[5]. Ces fonctionnalités peuvent cependant être désactivées par paramétrage, bien que cela ne soit pas très clairement documenté [6]. Virtual PC utilise quant à lui un mécanisme à base d'instructions processeurs illégales détectées par l'hyperviseur et déclenchant des services spécifiques. La liste complète des fonctions ne semble pas publique à ce jour. D'autres produits de virtualisation implémentent ce type de mécanismes[3].

Problèmes d'implémentation. La complexité de développement d'un virtualiseur reste la principale source de vulnérabilités concernant le cloisonnement. Deux sous systèmes sont particulièrement sensibles :

- Le système de partage des ressources matérielles comprenant la gestion mémoire, l'exécution des instructions, l'ordonnancement, etc.
- Les mécanismes d'émulation ou de partage des périphériques.

Une analyse réalisée en 2007 par Tavis Ormandy sur une sélection de virtualiseurs du marché[7] a ainsi mise en évidence un grand nombre de vulnérabilités causant des dénis de services et des compromissions des OS hôtes. Les outils de fuzzing relativement élémentaires utilisés et le nombre de problèmes rencontrés n'invitent pas à la confiance. Il faut cependant souligner que les virtualiseurs analysés ont été uniquement de type II, par conception plus vulnérables.

D'autres travaux d'analyse ont ensuite été menés impliquant la sécurité de systèmes de type I comme XEN 3 ou VMware ESX[8]. Ces problèmes mettent en évidence que les hyperviseurs de type 1 ne sont pas à l'abri de problèmes de sécurité et soulignent la criticité de l'OS privilégié[9].

[3] On se reportera à l'article de Peter Ferrie pour de plus amples détails[4].

Problèmes de conception. Enfin, même sans problèmes d'implémentation, on peut déplorer un certain nombre de choix architecturaux qui sont dommageables au niveau global de sécurité.

Par exemple, des travaux menés récemment au CELAR sur le virtualiseur XEN ont permis de mettre en évidence la possibilité d'établir un canal caché entre instances en contournant la politique de sécurité mise en place. XEN maintient ainsi en mémoire une table de conversion entre adresses mémoire accessible et en partie modifiable par toutes les instances. Il a été possible de développer une preuve de concept mettant en œuvre ce canal de communication. Le fait que cette table soit accessible à tout le monde est clairement un choix d'implémentation privilégiant les performances à la sécurité, la solution consistant à gérer une table par instance et de commuter la table en même temps que les instances étant coûteuse en performances.

Cette vulnérabilité avait déjà été évoquée sur la mailing-list de XEN mais son exploitation avait été jugée compliquée. Nous avons été surpris que la publication d'une preuve de concept permettant d'exploiter ce canal de communication n'ait pas fait réagir plus que cela les développeurs[4]. Bien que l'exploitation de ce canal caché implique d'avoir corrompu les deux instances désirant communiquer, il est quand même possible par ce biais d'établir un canal de communication au débit important et en contournant toute politique de sécurité mise en place par ailleurs. Des contacts directs avec les auteurs du mail initial ont montrés qu'ils étaient cependant toujours en train de travailler sur ces problèmes de sécurisation.

Toujours concernant ces aspects architecture, il existe une tendance actuelle visant à intégrer de plus en plus de fonctions au sein de l'hyperviseur. Par exemple, la taille de l'hyperviseur XEN n'a cessé d'augmenter au fil des versions pour atteindre aujourd'hui plus de 300000 lignes de code, pour entre 50000 et 100000 pour Hyper-V par exemple[9]. Cette croissance augmente mécaniquement les risques de problèmes de sécurité.

Les divers mécanismes de communication natifs entre instances décrits dans le paragraphe sur le cloisonnement sont enfin autant d'exemples de fonctionnalités qui ne devraient pas être implémentées sur un hyperviseur sécurisé. De même, l'hyperviseur XEN fournit une API de paravirtualisation relativement riche et utilisée au final uniquement en partie par les OS portés. Une bonne pratique de sécurité aurait été de réduire cette API au strict minimum nécessaire au fonctionnement d'un OS donné, afin de limiter la surface d'attaque (sachant que justement cette API est un élément particulièrement sensible puisque directement accessible par une instance corrompue).

4.4 Point sur la sécurité matérielle

La sécurité matérielle est actuellement un des problèmes majeurs dans la réalisation d'un hyperviseur sécurisé. Un certain nombre de travaux ont ainsi mis en évidence l'impact potentiellement désastreux que peut avoir un problème au niveau du matériel sur le niveau de sécurité d'un hyperviseur ou de manière plus générale d'un OS.

Problématique du DMA. La problématique du DMA est ainsi édifiante. Cette fonctionnalité permet à un périphérique matériel d'accéder directement à la mémoire de la machine, sans passer par le processeur et donc en contournant les contrôles mis en place par la MMU du processeur. Deux types de problèmes se posent ainsi :

1. Un OS invité corrompu contrôle un périphérique supportant le DMA afin d'atteindre des zones mémoire qui lui sont interdites (autres instances, instance privilégiée, hyperviseur).

[4] `http://lists.xensource.com/archives/html/xen-devel/2008-07/msg01116.html`

Un hyperviseur sécurisé peut lutter contre cela en n'accordant aux instances invitées aucun droit d'accès direct au matériel et en se plaçant systématiquement en coupure. il joue alors un rôle de proxy pour les I/O et il lui revient de filtrer et réécrire correctement les appels pour limiter les vulnérabilités [11] [12]. Ce filtrage ne peut cependant dans l'absolu pas être parfait. Ce type d'attaque a par exemple été utilisé récemment par Rafal Wojtczuk pour modifier un hyperviseur XEN en mémoire en utilisant les fonctionnalités d'une carte réseau et d'un disque dur [9]. L'attaque se déroulait dans ce cas cependant depuis le domaine 0, ce qui est quand même une hypothèse de départ extrêmement forte car le domaine 0 dispose d'un accès direct aux périphériques.

2. Un périphérique bogué ou piégé accède de sa propre initiative à des zones de mémoire normalement cloisonnées et effectue des opérations illicites. Lutter contre ce type de problèmes n'est pas possible sans une fonctionnalité nouvelle au niveau du chipset : l'IOMMU. Cette fonctionnalité additionnelle située dans le chipset permet de contrôler les accès d'un périphérique donné à la mémoire et permet à l'hyperviseur d'assurer un cloisonnement correct entre les instances et lui même vis à vis des périphériques. En outre, cela simplifie également la gestion des périphériques et permettrait même d'accorder à une instance les droits d'accès direct à un périphérique donné, à la condition que le périphérique soit dédié à cette instance (par exemple une carte graphique 3D par instance).

On n'oubliera pas enfin que certains bus de communication (par exemple FireWire) disposent nativement du support du DMA et qu'il convient pour plus de sécurité au niveau des périphériques amovibles de désactiver ce support [5].

Sécurité du processeur. Le processeur reste l'élément le plus critique et est à ce titre le composant envers lequel il est nécessaire d'avoir le niveau de confiance le plus élevé. L'exploitation d'une lacune au niveau du processeur peut ainsi rapidement avoir un effet désastreux sur la sécurité d'un mécanisme de virtualisation [11]. Pourtant, il demeure un certain nombre de points préoccupants quant au niveau de confiance que l'on peut réellement en attendre :

- Des bogues processeurs sont régulièrement publiés, et ces bogues peuvent être critiques d'un point de vue sécurité. Un mode de mise à jour du microcode des processeurs est bien présent mais reste peu documenté. Des mécanismes sont ainsi prévus pour pouvoir corriger d'éventuelles erreurs de conception des processeurs après coup (par exemple le bug TLB de l'architecture AMD K10 dont la correction cause une baisse conséquente des performances). Ces mécanismes sont habituellement utilisés par les BIOS des cartes mères pour remettre à niveau un processeur, cette mise à jour n'étant pas permanente. Soulignons que ces mécanismes sont peu documentés et que des essais de modification de microcode ont déjà été réalisés sur AMD K8 (qui apparemment ne protégeait même pas en intégrité son microcode, contrairement aux puces Intel). L'impact en termes de sécurité reste encore relativement indéterminé : tous les bugs sont-ils biens corrigés ? Le BIOS d'une machine donnée inclue t-il bien tous les correctifs ? On est en droit de se poser des questions...
- Des instructions non documentées peuvent être présentes. Ce cas c'est déjà produit par le passé sur les pentiums, où des instructions supplémentaires avaient été mises en évidence et sont depuis documentées.
- Des modes de fonctionnement du processeur peuvent poser problème, comme par exemple le mode de maintenance SMM. Quand le processeur passe dans ce mode, il dispose d'un accès complet à l'ensemble de la mémoire ce qui est évidemment problématique. Le fondeur

[5] On se reportera à la présentation "voyage au cœur de la mémoire" où l'auteur utilise un portable sous linux en se faisant passer pour un iPod pour accéder directement à la mémoire d'un poste Windows [13]

a cependant prévu des mécanismes de sécurité visant à protéger le code de maintenance exécuté lors du passage en mode SMM. Ce code, situé dans une zone de mémoire rendue inaccessible après le BIOS, ne peut normalement plus être modifié par la suite. Des techniques ont cependant été publiées pour contourner ce mécanisme de sécurité, qui sont pris en compte au fur et à mesure par les éditeurs de BIOS (voir à ce sujet l'attaque SMM décrite par Loic Duflot en 2006[12], plus récemment celle de Sun Bing nécessitant un reboot pour fonctionner [14], ou enfin la technique récente basée sur du remapping mémoire sur chipset Q35 [9]).

Le chipset. Le chipset fournit également un certain nombre de fonctionnalités qui peuvent être détournées de leur but initial à des fins hostiles. Par exemple en 2006, l'utilisation de la fonctionnalité d'ouverture graphique [11] avait permis à Loïc Duflot de modifier à la volée le secure level d'un système OpenBSD. Plus récemment c'est une autre propriété du chipset Intel Q35 qui a été utilisée pour contourner les protections mises en place par un hyperviseur XEN malgré l'utilisation d'un IOMMU. Dans les deux cas il existait pourtant des mécanismes de sécurité destinés à empêcher l'exploitation illicite de ces fonctionnalités, malheureusement pas utilisées par les éditeurs de BIOS. D'autres fonctionnalités seraient susceptibles de causer de tels problèmes de sécurité (fonctions ACPI par exemple).

Les périphériques. Outre les problématiques de DMA, les périphériques peuvent également fournir des fonctionnalités plus ou moins bien documentées dont l'impact en termes de sécurité est encore indéterminé. Ainsi par exemple les disques durs proposent un certain nombre de commandes constructeur peu documentées [15]. Ceci est également valable avec la plupart des autres périphériques (cartes réseau, cartes graphiques, etc ...).

Zones de mémoire rémanente. Il existe de nombreuses zones de mémoire rémanentes au sein d'un poste informatique qui peuvent causer des fuites d'informations entre instances d'un système virtualisé par l'établissement de canaux cachés. Par exemple certains périphériques disposent désormais de zones de mémoire internes et de protocoles permettant d'y accéder, le plus souvent pas ou peu documentés : écrans, claviers, souris, etc. Dans le même esprit, la problématique de la rémanence de la mémoire RAM mise en évidence récemment peut nous poser des problèmes si le virtualiseur ne prends pas garde à bien vider la mémoire entre deux redémarrages.

Les firmwares. Les différents firmwares d'une machine sont enfin un sujet de préoccupation. Outre le BIOS de la carte mère, de nombreux périphériques disposent également de leur propre firmware que le BIOS principal va appeler tour à tour. Ces firmwares sont les premiers programmes s'exécutant sur la plateforme avant de donner la main au chargeur du virtualiseur. Or leur fonctionnement interne est jalousement gardé secret par les éditeurs et très peu documenté. De plus ces BIOS peuvent être mis à jour et quelques travaux de reverse engineering et de modification d'un BIOS ont été publiés [16]. Cette mise à jour se fait habituellement par un port d'I/O spécifique permettant le plus souvent de flasher le BIOS depuis le système d'exploitation, on peut donc espérer que le mécanisme de virtualisation empêche une instance d'accéder par ce biais au BIOS (mais qui peut garantir que tous les BIOS déployés se flashent de cette manière ?). Au final le BIOS est donc à la fois un élément obscur et peu documenté, modifiable, candidat idéal pour piéger une machine discrètement car s'exécutant avant l'application de virtualisation, et en lequel on est obligé d'avoir une confiance quasi aveugle. Et il n'est pas dit que le remplaçant du BIOS, l'UEFI, améliorera les

choses. Plus ouvert et documenté, il est également beaucoup plus complexe (c'est quasiment un véritable système d'exploitation). Au final le niveau de confiance que l'on peut apporter à cet élément crucial est quand même relativement limité.

4.5 Problématique de la zone de confiance

Le plus souvent, les mécanismes de virtualisation ont ainsi besoin d'un OS privilégié chargé de piloter le mécanisme de virtualisation. Cet OS peut être un OS hôte au sein duquel le mécanisme de virtualisation est installé (Type II) ou un OS virtualisé disposant de droits étendus pour paramétrer le mécanisme de virtualisation (type I). Ces instances privilégiées hébergent la plupart du temps également les drivers pilotant les périphériques de la plateforme.

Dans les deux cas, cet OS doit avoir le niveau de confiance le plus élevé possible car il est particulièrement critique. Cette zone de confiance pourra permettre également d'implémenter des mécanismes de sécurité sensibles. L'amélioration du niveau de confiance de cet OS privilégié peut se faire selon plusieurs pistes :

- Réutilisation d'un micronoyau minimaliste implémentant uniquement les fonctions souhaitées ;
- Réutilisation d'un OS complet (Linux, Windows) que l'on aura durci via une suppression des fonctionnalités inutiles, un paramétrage adapté et l'ajout de mécanismes de sécurité supplémentaires (cloisonnement, patchs sécurité, modèle d'accès MAC, etc.).

A souligner que de nombreux travaux sont actuellement réalisés dans le cadre du projet XEN afin de "découper" ce domaine privilégié. L'idée est d'isoler certaines fonctions sensibles pour limiter l'impact d'une éventuelle faille à ce niveau (isolation des drivers dans des domaines spécifiques par exemple).

4.6 Rootkits à base de mécanismes de virtualisation : quand la virtualisation devient une menace

Un nouveau type de malwares. La présentation "Subverting Vista Kernel For Fun And Profit" [10] donnée en juillet 2006 par Joanna Rutkowska durant la conférence SyScan 2006 puis en août à la Black Hat 2006 a eu un fort retentissement. Outre le fait que cette présentation décrit une méthode pour contourner le mécanisme de protection du noyau par signature de code de Vista sur architecture x86-64, cette présentation a surtout permis de décrire les principes d'une nouvelle famille de rootkits basés sur l'utilisation de fonctionnalités de virtualisation ainsi que de présenter une preuve de concept sous la forme du rootkit BluePill. L'idée principale de cette famille de rootkits est que, au lieu d'essayer de se camoufler à l'intérieur d'un OS et donc fatalement de laisser des traces qui pourront être repérées, il est possible d'éviter la détection en se plaçant à l'extérieur de l'OS comme le fait un hyperviseur licite. Un tel type de malware devient ainsi extrêmement difficile à détecter pour un antivirus s'exécutant au sein de l'OS attaqué, étant donné qu'il ne modifie rien au sein de l'OS infecté[6].

Historiquement, BluePill n'est cependant pas le premier malware du genre. Des chercheurs de l'université du Michigan avaient en effet présenté un peu plus tôt dans l'année leur preuve de concept Sub Virt [17], basée sur la réutilisation des virtualiseurs logiciels VMware ou Virtual PC. L'idée est ici d'installer un nouvel OS sur le poste attaqué qui servira d'OS hôte, et de présenter à l'utilisateur son OS originel virtualisé. Ce mode de fonctionnement à base de

[6] On se reportera à l'article "Security Challenges in Virtualized Environments" [8] où Joanna Rutkowska présente une classification des malwares et qui introduit ce nouveau type de malwares se plaçant en dehors du système d'exploitation.

virtualiseur du commerce en fait cependant un rootkit peu discret et facile à détecter (pertes de performances, obligation de démarrer l'OS hôte avant l'OS invité, techniques de détection usuelles applicables).

Basé sur l'utilisation des fonctionnalités de virtualisation matérielles AMD-V, BluePill est beaucoup plus discret. Il peut ainsi virtualiser un OS à la volée (sans reboot), est plus difficilement détectable par les techniques de détection usuelles et provoque peu de pertes de performances. À souligner enfin qu'un équivalent à BluePill nommé Vitriol [18] sur architecture Intel VT a également été présenté lors de la BlackHat 2006, au fonctionnement très similaire.

Techniques de défense adaptées. La lutte contre ce type de malwares n'est pas aisée car ils ne laissent pas de traces au sein de l'OS lui même. La détection est cependant possible depuis l'OS attaqué selon les méthodes d'analyse temporelle déjà décrites dans cet article, mais deviendra de plus en plus difficile au fur et à mesure que les systèmes d'exploitation utiliseront eux même la virtualisation de manière native. Pour lutter contre ces malwares, des mécanismes de détection commencent ainsi à être proposés utilisant eux aussi les mécanismes de virtualisation matériels pour se placer hors du système d'exploitation et assurer sa surveillance et sa défense. Le mécanisme prenant le premier la main sur la plateforme contrôlant cette même plateforme, on assiste donc actuellement à une sorte de course entre attaquant et défenseur pour être le premier à s'exécuter. Dans cette course, on commence à voir apparaitre des solutions se lançant directement au niveau du BIOS ou de l'UEFI, du chipset ou même s'exécutant en mode SMM [19].

Enfin, l'intégrité de la machine dès le démarrage est un élément important de la confiance. À cet effet, l'utilisation de technologies de vérification d'intégrité basées sur une puce TPM peut permettre d'apporter un meilleur niveau de confiance. Ces mécanismes permettent de mettre en place une chaine de confiance dès le démarrage du poste et de s'assurer de l'intégrité des logiciels démarrant sur la plateforme, et donc dans notre cas du virtualiseur. Intel avec sa technologie TXT et AMD avec sa technologie SVM ont intégré ces fonctionnalités sur leurs plateformes professionnelles et les virtualiseurs commencent à utiliser ces mécanismes (XEN, Hyper-V).

5 Utilisations dans des architectures de sécurité

De nombreuses utilisations de la virtualisation sont possibles dans des architectures de sécurité.

5.1 Durcissement d'un système d'exploitation.

Dans le contexte d'un poste informatique de confiance, la virtualisation peut permettre de renforcer la sécurité d'un système d'exploitation en plaçant hors d'atteinte un certain nombre de fonctionnalités critiques d'un point de vue sécurité, afin de protéger ces fonctionnalités d'une compromission du système d'exploitation principal. Les fonctions à protéger peuvent être des modules d'antivirus, de vérification d'intégrité ou de détection d'intrusion. Des mécanismes cryptographiques ou d'entrée de données utilisateurs sensibles (code PIN, mot de passe) peuvent également être implémentés directement dans cette zone de confiance. On peut envisager à ce propos deux approches :

- La première est l'intégration de ces fonctions au sein même d'un hyperviseur réduit dédié sécurité. Cette approche est discutable car cela cause une augmentation de la taille de l'hyperviseur. De plus cela contraint à intégrer l'ensemble des drivers au sein même de l'hyperviseur pour se prémunir des attaques utilisant les fonctionnalités matérielles, ce qui n'est pas souhaitable.
- La deuxième est l'utilisation d'une zone de confiance formée par un système d'exploitation dédié et minimaliste, faisant fonctionner les modules de sécurité et accueillant les drivers. Cette option parait la plus réaliste.

5.2 Poste client multiniveau.

En poussant le concept précédent un peu plus loin, il est également possible d'envisager un poste multiniveau en conservant un hyperviseur dédié sécurité et une zone de confiance, et en multipliant les instances invitées sur le poste. On associe alors une instance invitée par réseau de niveau de sécurité distinct. On peut ainsi par exemple imaginer une instance dédiée à un réseau de confiance interne, une autre instance dédiée à un réseau privilégié avec des partenaires, et une troisième instance dédiée Internet. Des exemples de tels types de systèmes sont déjà disponibles ou en cours de développement :

- NetTop est un client multiniveau US réalisé par HP, basé sur l'utilisation d'un OS hôte utilisant les mécanismes de sécurité SE Linux et assurant comme mécanisme de virtualisation VMware (hyperviseur de type II) ;
- Integrity PC est également un client multiniveau US proposé par la société GreenHills. Ce système est basé sur un système d'exploitation temps réel (Integrity RTOS), déjà utilisé dans des applications aéronautiques et en cours d'évaluation EAL6+ ;
- SINA Virtual Workstation est un client multiniveau Allemand réalisé par la société Secunet. Basé sur un Linux durcis, il utilise VMware en mécanisme de virtualisation ;
- SINAPSE est un prototype de poste client multiniveau Français développé par la société Bertin pour la DGA. Ce poste est basé sur un mécanisme de virtualisation de type I propriétaire et un OS privilégié minimal dérivé de ChorusOS. Le tout est en cours d'évaluation au niveau EAL 5.
- Le groupement OpenTC, qui vise à promouvoir l'informatique de confiance, propose deux prototypes de postes client multiniveau basés sur les virtualiseurs XEN ou Fiasco/L4 Linux. Ces démonstrateurs protègent également leur intégrité grâce à l'utilisation de puces TPM.

5.3 Durcissement de serveurs

D'un point de vue serveur, la virtualisation peut servir à mettre en place un cloisonnement robuste entre différents services pour éviter une compromission complète en cas d'exploitation d'une faille dans un des services. Cette approche permet en première analyse de renforcer le niveau de sécurité par rapport à simple confinement réalisé via des mécanismes d'isolation (Jail, V/Server). On peut estimer que le niveau d'assurance est cependant moindre que si l'on avait disposé de plusieurs serveurs physiques, les vulnérabilités propres aux mécanismes de virtualisation s'ajoutant aux vulnérabilités standards des serveurs.

5.4 Mise en place d'architectures de sécurité virtuelles

En allant plus loin que le durcissement d'une plateforme informatique cliente ou serveur, il existe une tendance actuelle consistant à virtualiser tout ou partie des composants d'une

DMZ [20]. L'idée est ici de gagner en flexibilité et en granularité de la DMZ en définissant une architecture de DMZ virtuelle complexe que l'on n'aurait pas eu les moyens d'acquérir en version complètement physique. Il est ainsi envisageable de virtualiser chaque zone de la DMZ sur un serveur virtualisé différent, ou de virtualiser toutes les zones sur un même serveur disposant de plusieurs cartes réseau et en conservant une segmentation réseau physique (commutateurs, routeurs, pare-feux), ou enfin de virtualiser l'ensemble de l'architecture y comprit le réseau. On commence ainsi à voir apparaitre sur le marché des offres d'appliances de sécurité virtuelles destinées à la constitution de ces DMZ virtuelles. Si la démarche en termes de sécurité peut paraître louable, on se retrouve entièrement tributaire du niveau de sécurité de la solution de virtualisation. Au vu des limitations exposées dans cet article, il est au final recommandé de limiter l'utilisation de telles solutions complètement virtualisées et de conserver au minimum une segmentation physique complète.

6 Conclusion

En conclusion, l'utilisation de la virtualisation pour améliorer le niveau de sécurité d'une plateforme informatique semble pleine de promesses. En particulier, la virtualisation apporte une fonction de cloisonnement local proche du matériel et plus facile à évaluer que les mécanismes de cloisonnement présents au sein même de l'OS. Une architecture de type I dont l'hyperviseur fonctionne directement au dessus du matériel sera cependant à privilégier. On soulignera que, même dans ce cas, le domaine privilégié pilotant la virtualisation reste un domaine critique d'un point de vue de la sécurité.

Ce constat doit cependant être tempéré par le fait que les hyperviseurs actuels n'ont pas réellement été développés avec un objectif de sécurité. Des choix de conception discutables ainsi que de nombreuses vulnérabilités mises en évidence montrent que fa plupart des hyperviseurs actuels ne sont pas encore à même de proposer un niveau de sécurité élevé. Ceci est d'autant plus vrai que des travaux récents montrent que des vulnérabilités au niveau du matériel peuvent permettre de contourner le cloisonnement mis en place par un hyperviseur et rapidement tout remettre en cause. Ceci contraint à devoir accorder un niveau de confiance relativement important à sa plateforme matérielle, niveau de confiance qui ne parait pas vraiment justifié au vu du nombre de problèmes qui commencent à être remontés à ce niveau.

La virtualisation peut donc permettre le fonctionnement de plusieurs instances de systèmes d'exploitation sur le même poste physique, afin de durcir postes client ou serveurs, de réaliser des postes clients multiniveau ou même de virtualiser des architectures de DMZ. Au vu des limitations actuelles des technologies considérées, on prendra cependant garde à ne pas se baser sur ce mécanisme de cloisonnement comme unique barrière de sécurité et à lui associer d'autres mesures. On pourra par exemple durcir les OS virtualisés ou même ne pas virtualiser l'ensemble d'une architecture et conserver un minimum de segmentation physique (machines et réseaux) aux endroits vraiment critiques afin de limiter l'impact d'une défaillance du mécanisme de virtualisation.

Références

1. G. Popeck & R. Goldberg : Formal Requirements for Virtualizable 3rd Generation Architectures - Communications of the A.C.M., 1974
2. John Scott Robin & Cynthia E. Irvine : Analysis of the Intel Pentium's Ability to Support a Secure Virtual Machine Monitor - U.S. Air Force & Naval Postgraduate School, 2000

3. Hypervisor Functional Specification Draft version 0.98 - http://www.microsoft.com
4. Peter Ferrie : Attacks on virtual machine emulators v2 - Symantec Advanced Threath Research, avril 2007.
5. Ken Kato : VMware Backdoor I/O Port - http://chitchat.at.infoseek.co.jp/vmware/backdoor.html
6. Sandbarrow.com - VMX File parameters - http://sanbarrow.com/vmx/vmxadvanced.html\#isolationtools
7. Tavis Ormandy : An Empirical Study into the Security Exposure to Hosts of Hostile Virtualized Environments, conférence CanSecWest 2007
8. Joanna Rutkowska : Security Challenges in Virtualized Environnements - Invisible Things Labs, 2007
9. Joanna Rutkowska, Rafal Wojtczut & Alexander Tereshkin : Xen Owning Trilogy - Conférences Black Hat USA 2008, août 2008
10. Joanna Rutkowska : Subverting Vista Kernel For Fun And Profit - Conférences SyScan et Blak Hat 2006, juillet 2006.
11. Loic Duflot : Bogues ou piégeages des processeurs, quelles conséquences sur la sécurité ? - Conférence SSTIC 2008
12. Loic Duflot, Daniel Etiemble, Olivier Grumelard : Utiliser les fonctionnalités des cartes mères ou des processeurs pour contourner les mécanismes de sécurité des systèmes d'exploitation - Conférence SSTIC 2006
13. Damien Aumaitre : Voyage au cœur de la mémoire - Conférences SSTIC 2008
14. Sun Bing - BIOS boot hijacking And VMware Vulnérabilities Digging - POC 2007 Seoul Korea, novembre 2007
15. Laurent Dupuy : Indiscrétions et "zones constructeurs" des disques durs - Conférence SSTIC 2007
16. Scythale : Hacking Deeper in the System - Phrack n°64 - juillet 2007
17. Samuel T. King et Peter M. Chen : SubVirt - implementing malwares with virtual machines, université du Michigan, conférence IEEE Security and Privacy 2006
18. Dino A. Dai Zovi : Hardware Virtualization Rootkits - Conférence Black Hat USA 2006
19. Don Bailey : UEFI Hypervisors - Winning the Race to the Bare Metal - Conférence Black Hat 2008
20. Julien Raeis & Nicolas Collignon : VMware et sécurité - OSSIR Groupe SUR - Cabinet HSC - juillet 2008

Composant cryptographique TPM
Retours d'expériences et perspectives

Frédéric Rémi et Goulven Guiheux

Laboratoire d'évaluation d'AMOSSYS
Espace performance Bât M1 35760 SAINT GREGOIRE
{frederic.remi,goulven.guiheux}@amossys.fr

Résumé Cet article à vocation didactique présente les fonctionnalités mises à disposition des applications par un composant cryptographique TPM. Les services de haut-niveaux (mesure du poste client, attestation distante, stockage sécurisé, ...) utilisant le composant TPM sont détaillés. Un retour d'expérience sur la conformité et l'efficacité des services et des principales applications (notamment la virtualisation) est effectué via le déroulement d'un plan de tests sur deux puces représentatives du marché. Pour conclure, nous présentons les principales perspectives et problématiques offertes par cette technologie pour la sécurisation des architectures de type PC.

Mots-clef : TPM, TSS, TCG, Trusted Computing, PCR, SRK, EK, CRTM, DRTM, SRTM, vTPM, Scellement, Attestation distante, IMA, Trousers.

1 Introduction

Le manque de confiance dans la sécurité des architectures de plateformes ouvertes de type PC a poussé en octobre 1999, IBM, Intel, Microsoft, Compaq et HP à se regrouper au sein de La « Trusted Computing Platform Alliance » ou littéralement « alliance pour une informatique de confiance ». Aujourd'hui, ce sont plus de 130 sociétés qui ont rejoint l'alliance devenue depuis avril 2003 le TCG ou « Trusted Computing Group ».
Le TCG spécifie notamment l'architecture du composant cryptographique TPM (Trusted Platform Module) dont l'objectif principal est d'améliorer la sécurité des plateformes de type PC. Cette amélioration sécuritaire repose sur la spécification de services de sécurité haut-niveaux utilisant les fonctionnalités du TPM (mesure du poste client, attestation distante, stockage sécurisé dépendant de l'état de la plateforme).
Le principe de base du TPM est d'offrir aux utilisateurs un composant physique de confiance sur le poste client. Ce composant met à disposition de l'utilisateur des fonctions cryptographiques de base (génération d'aléa, de clefs, de signatures électroniques, de hachage), de stockage sécurisé et d'agrégation de mesures. La confiance affichée provient alors de la nature matérielle du composant. Le pilotage du composant TPM est quant à lui réalisé via une architecture logicielle dédiée, la TSS pour TCG Software Stack.
L'objectif du TCG est de fournir un service attestant de l'intégrité d'une plateforme reposant sur cette architecture matérielle et logicielle. Cette architecture permet de mesurer les informations jugées sensibles (fichiers exécutables, librairies, pilotes, noyau du système, clefs d'activation, fichiers de configurations, données personnelles, ...). Ce service est décomposé en un service de mesure de l'intégrité du poste client et un service d'attestation distante.

L'attestation distante se charge de communiquer via un protocole spécifique les mesures effectuées à un tiers de confiance qui les valide ou non. Pour cette raison, le TCG est perçu par ses détracteurs comme une menace sérieuse pour les libertés individuelles. En effet, le fait que chaque programme soit accompagné d'une signature validée par un tiers restreint l'utilisation de logiciels libres. En trame de fond se cache la possibilité pour les promoteurs du TCG d'empêcher toute concurrence sur leur plateforme sécurisée en bloquant l'installation de logiciels tiers. Cette caractéristique leur a d'ailleurs valu le surnom de Treacherous Computing (Informatique déloyale), en opposition au nom officiel Trusted Computing (Informatique de confiance).

2 Présentation du composant TPM

2.1 Architecture matérielle

Le TPM est un composant passif de type carte à puce interagissant avec sa pile utilisatrice selon un modèle challenge-réponse. Le composant est de préférence soudé à la carte mère et possède un accès bidirectionnel avec le CPU. L'accès se fait par le biais d'un port bas débit LPC d'une bande passante variant de 256 Mo/s à 1 Go/s selon les modèles, géré par le « South Bridge » . Le TPM est composé de plusieurs modules décrits fonctionnellement dans les paragraphes suivants.

Module Input/Output (I/O). Le module I/O gère le flux d'information véhiculé par le bus de communication. Il met en œuvre le protocole de codage/décodage des flux entrants et sortants et route les données vers les modules appropriés. Le composant TPM est relié à un port bas débit LPC (Low Pin Count). Par son positionnement, ce composant se distingue naturellement des cartes à puces ou autres clefs USB qui interagissent avec la plateforme matérielle après le chargement de l'OS.

Module Non-Volatile Storage. Le module de mémoire non-volatile permet d'assurer le stockage permanent des données suivantes en mémoire.

- Endorsement Key (EK). Paire de clefs asymétriques générée par le fabriquant de la puce TPM pour son identification en phase de personnalisation du composant. Au premier démarrage du composant, un protocole d'activation du composant est initié entre le fabriquant et le poste client. La bonne exécution de ce protocole entraîne la certification par le fabriquant de la partie publique de la clef EK. Ce certificat devient la carte d'identité du composant TPM, il fournit la preuve que le TPM est authentique, i.e. que ce n'est pas une contrefaçon ou une virtualisation logicielle du composant matériel ;
- Storage Root Key (SRK). Paire de clef asymétrique générée à l'exécution de la commande `TPM_TakeOwnership` (cf ci-dessous). Elle joue le rôle de clef maître ou racine dans un système de hiérarchie des clefs classique. Seule sa partie privée est stockée dans une partie protégée de la mémoire non volatile. La partie publique de la clef est utilisée pour le chiffrement de données ou de clefs filles.
- Owner Authorization Data. À l'exécution de la commande `TPM_TakeOwnership` par le propriétaire du poste client, un condensé de 160 bits est créé et stocké en mémoire. Cette valeur constitue un mot de passe commun entre le composant et le propriétaire de la plateforme. Il permet d'assurer au propriétaire l'usage exclusif de capacités de sécurité du composant TPM. Malgré l'usage du terme `Authorization Data`, ce mot de passe partagé est utilisé pour assurer un usage exclusif du composant par le propriétaire légitime.

Module Volatile Storage. Le module de mémoire volatile permet d'assurer le stockage des données temporaires.

Module Plateform Configuration Register. Les mémoires PCR sont des registres de 160 bits permettant de stocker l'état d'une plateforme. Ils sont utilisés pour agréger les mesures utilisées pour attester de l'intégrité du poste client. Le TCG spécifie un minimum de 16 registres pour la norme 1.1 et de 24 registres pour la norme 1.2 dont les 8 premiers (0-7) sont réservés à un usage interne du TPM. Les autres registres sont réservés aux applications (*i.e.* chargeur d'amorce, noyau, librairies système, drivers, applications).

Module Attestation Identity Keys (AIKs). Ce module optionnel permet le stockage des clefs AIK. Les clefs AIK sont des bi-clefs utilisées lors du protocole d'attestation de la plateforme. Nous décrivons leur utilité dans le paragraphe. Les AIK peuvent être tout aussi bien stockées sur un système de stockage externe au TPM.

Modules Program code and Execution Engine. Le module Program code contient le firmware du composant TPM. Le module Execution Engine constitue le CPU du TPM et à ce titre exécute les commandes envoyées par le module I/O en interprétant le micro code du firmware.

Module Random Generator (RNG). Ce module implémente un générateur d'aléa physique. Cet aléa est utilisé pour la génération des clefs, des vecteurs d'initialisation et des challenges aléatoires pour les protocoles cryptographiques.

Module SHA-Engine. Ce module implémente la fonction de hachage cryptographique SHA-1. Le TCG étant un centre de normalisation, ce module devrait évoluer dès la standardisation par le NIST d'un nouvel algorithme de hachage cryptographique en remplacement de SHA-1.

Module HMAC Engine. Ce module implémente la fonction de hachage à clef HMAC. La taille des clefs est de 20 octets et la taille des blocs est de 64 octets. La fonction supporte le calcul du HMAC selon la RFC 2104.

Module Key Generation. Le module de génération de clefs permet de créer des clefs symétriques et asymétriques. Le module de génération supporte des clefs RSA jusqu'à 2048 bits (512 à 2048 bits).

Module RSA Engine. L'algorithme RSA est utilisé à la fois en mode signature et en mode chiffrement. Le TCG respecte le standard PKCS#1 qui fournit les détails des spécifications pour la signature, le chiffrement et le formatage des données RSA.

Module Opt-In. Le module Opt-in implémente la politique de configuration du composant TPM. Plusieurs états sont définis « Enabled/Disabled » , « Active/Inactive » et « Owned/Unowned » .

Module Power Detection. Ce module permet de gérer les états d'alimentation du TPM en conjonction de ceux de la plateforme. À tout moment, le TPM doit connaître les changements d'état de la plateforme. Certaines commandes du TPM peuvent être restreintes selon la valeur de ces états.

Module de chiffrement symétrique. Le TCG spécifie le générateur de pseudo-aléa MGF1 de PKCS#1v2.1 en mode chiffrement par flot pour le service de confidentialité des sessions d'authentification et de transport. Concernant le service de confidentialité des données internes, le composant TPM peut également utiliser un algorithme de chiffrement symétrique mais le choix revient au constructeur. On remarquera que le TCG ne spécifie aucun algorithme symétrique par flot ou par bloc pour le chiffrement de données utilisateur.

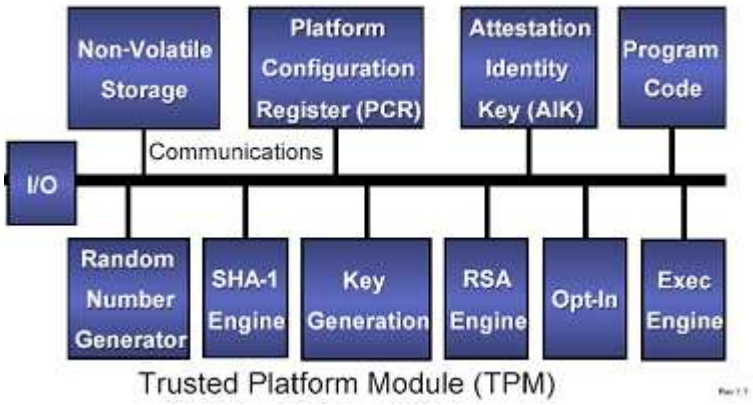


FIG. 1: Architecture matérielle du composant TPM

2.2 Services intrinsèques au composant

Les services offerts nativement par le composant sont les suivants :

- *Unicité de la plateforme.* Chaque TPM est associé à une clef RSA unique appelée Endorsement Key (EK). Cette clef est générée par le fabriquant de la carte mère en phase de personnalisation du TPM. La partie publique de cette clef est certifiée par une AC associant à chaque TPM un certificat unique. Cette clef ne doit jamais être communiquée à l'extérieur du TPM ;
- *Stockage sécurisé.* Le TPM intègre un service de protection des données en confidentialité. Ce service repose sur une hiérarchie de clefs. Chaque clef est protégée par une clef de niveau supérieur. La racine de cette hiérarchie est constituée par la clef SRK (Storage Root Key). Cette clef est stockée à l'intérieur du TPM, tandis que les autres clefs sont stockées sur un support externe (disque dur par exemple). La clef SRK est générée par le TPM lors de la prise de possession du TPM en utilisant la commande `TPM_TakeOwnership`.

- *Scellement de données*[1]. Ce service est un mécanisme de stockage sécurisé dépendant de l'état de la plateforme. Le « descellement » d'une donnée n'est possible que si l'état des registres PCR au moment du scellement est identique à l'état des registres PCR au moment du descellement. Ce mécanisme permet de conditionner le descellement d'un élément par rapport à l'intégrité de la plateforme.
- *Opérations cryptographiques.* Le TPM met à disposition de l'utilisateur les mécanismes cryptographiques suivants :
  - Fonction de hachage SHA-1 ;
  - Génération de clefs, chiffrement et signature RSA bits conformes aux travaux du groupe de standardisation de l'IEEE P1363 ;
  - Génération d'aléa.
- *Extension des registres PCR.* Ce mécanisme est utilisé pour les services de mesure de la plateforme et d'attestation distante. Il permet d'agréger des mesures réalisées par des logiciels tiers dans les registres PCR du TPM. Le processus d'extension est une opération cryptographique simple consistant à concaténer la valeur courante du registre PCR avec la nouvelle mesure puis à hacher le tout :

$$PCR_{t+1}[i] \leftarrow \text{SHA-1}\Big(PCR_t[i]||M\Big) \tag{1}$$

  M désigne la mesure réalisée par une application logicielle. Selon le concept de mesure définit par le TCG, $M = \text{SHA-1(DATA)}$. La variable i désigne l'index du registre concerné par l'extension. Cette opération offre plusieurs avantages :
  - Il est calculatoirement impossible de trouver des collisions sur la valeur des registres pour deux mesures différentes ;
  - La mesure n'est pas commutative, i.e. deux mesures déséquencées produisent des résultats différents ;
  - L'opération permet de stocker un nombre illimité de mesures dans les registres PCR puisque le résultat est toujours une empreinte de 160 bits ;
  - L'écriture directe dans un registre PCR est impossible.
- *Contrôle d'accès et de droits.* La révision 1.2 du TPM spécifie le mécanisme de « localities ». Ce mécanisme permet notamment de contrer certaines attaques matérielles telles que l'attaque par réinitialisation du bus LPC. Cette attaque très simple applicable au TPM 1.1. est décrite sur le site de l'université américaine de Darthmouth[10]
  Une localité représente un niveau d'accès au TPM. Des commandes spécifiques envoyées sur le bus LPC permettent de préciser le niveau de localité. Suivant la valeur de ce niveau (variant de 0 à 4), certains objets tels que les registres PCR peuvent être accessibles en lecture, en écriture (fonction `extend`) et peuvent également être réinitialisables. Par ce principe, l'attaque mentionnée ci-avant échoue par la difficulté de forger ces nouvelles commandes sur le bus LPC. Le principe de localité sert également dans l'élaboration d'une chaîne de confiance dynamique (DRTM).[2]

2.3 Architecture logicielle et services associés

La TSS constitue une spécification logicielle fournissant un standard d'API permettant d'accéder aux fonctions de la puce TPM. Elle établit le lien entre les programmes ou le système

[1] L'intitulé scellement n'est pas conforme à l'usage, la fonction est ici un chiffrement conditionnée à l'état des registres PCR

[2] Cf. paragraphe 3.1

d'exploitation et le pilote de la puce. Les développeurs peuvent l'utiliser pour créer des applications interopérables fonctionnant avec le TPM. La version actuelle de la spécification de la TSS est en version 1.2 datant de mars 2007. Les objectifs de l'API TSS sont :

- d'apporter un point d'entrée pour les applications aux fonctionnalités du TPM ;
- de gérer les services du TPM ;
- d'assurer la gestion du composant (synchronisation, contrôle de flux, audit des requêtes, ...)

La figure 2 illustre l'architecture logicielle associée au TPM.

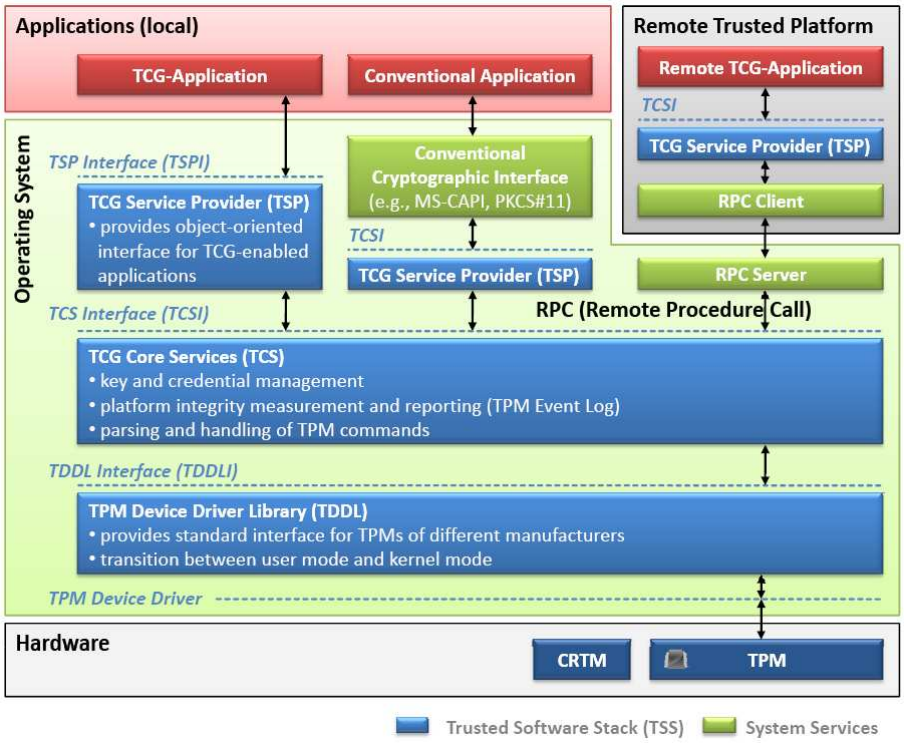


FIG. 2: Diagramme de la TSS

- Au niveau le plus bas, le composant TPM est accessible via le driver TPM (TDD) situé au même niveau que le noyau du système d'exploitation (ring 0). Ces drivers fournis par les constructeurs ne sont pas inclus dans la TSS ;
- Au niveau utilisateur, la TSS est composée de :
  - Un module de gestion de driver TPM (**TDDL**) qui constitue une couche d'abstraction aux drivers TPM permettant de développer des composants logiciels indépendants du driver TPM considéré. Il permet en outre le passage du mode utilisateur au mode noyau (TPM Device Driver). Le TPM est associé à un seul module TDDL. Ce module est initié et fournit par le fabriquant du TPM.
  - La couche des services principaux (**TCS**) fournit toutes les primitives du TPM et l'ensemble des services communs :

- Synchronisation des requêtes envoyées au TPM[3] ;
- Journalisation des évènements et des opérations ;
- Audit des évènements qui ont fait l'objet d'une mesure via le processus d'extension des registres ;
- Gestion des certificats de la plate-forme ;
- Gestion des informations d'authentification et d'identification.

Le TPM est associé à un seul module TCS. Ce module est exécuté comme un service système en mode utilisateur (daemon) et est utilisé par des applications de bas-niveaux implémentant des services spécifiques (middleware ou composant d'OS).

- Une couche de services de haut-niveau, la (**TSP**), utilisée par les applications souhaitant utiliser le TPM. Elle met à disposition de ces applications les services de la TCS en masquant les contraintes de gestion.

3 Services haut-niveaux

3.1 Mesure du poste client

La mesure du poste client est un service permettant de mesurer l'intégrité d'un poste client depuis la phase de boot. Ce mécanisme constitue la brique du service d'attestation distante qui se charge de vérifier les mesures réalisées.
La mesure de l'intégrité de la plateforme est réalisée en établissant une chaîne de confiance liant tous les éléments critiques de la configuration. Depuis le boot du poste client, chaque élément logiciel réalise une mesure de l'élément suivant avant de l'exécuter. En pratique, les éléments logiciels sont les suivants :
- Le BIOS ;
- Le gestionnaire de démarrage (Grub) ;
- Le système d'exploitation ;
- Les applications.

La mesure d'une donnée ou d'une application est réalisée par une fonction de hachage SHA-1. Cette mesure est à l'initiative des applications. L'ensemble des mesures successives forment une chaîne de confiance initiée par le CRTM (Core Root Trusted Measurement). Les spécifications 1.2 distinguent deux types de CRTM :
- Le S-CRTM (Static CRTM) : Il s'agit du code exécuté en premier sur le poste client au démarrage. Son objectif est de réaliser une mesure de lui-même et de l'élément logiciel suivant. Concrètement, sur un poste client, le S-CRTM représente l'ensemble des premiers octets du BIOS qui s'auto-mesure puis mesure le reste du BIOS.
- Le D-CRTM (Dynamic CRTM) : Il s'agit d'un élément logiciel qui permet de débuter une nouvelle chaine de confiance après le démarrage du poste client. Les chaînes de confiance respectivement initiées par le S-CRTM et le D-CRTM sont indépendantes. L'élément logiciel débutant la nouvelle chaîne de confiance est appelé « chargeur sécurisé » (secure loader). L'amorçage du D-CRTM est réalisé en plusieurs étapes :
  - Copie en mémoire du chargeur sécurisé ;
  - Réinitialisation sécurisée du processeur avec les instructions skinit (processeur AMD) ou senter (processeur Intel). Ces instructions permettent de réinitialiser le processeur et d'envoyer un signal protégé de reset au TPM. Le chargeur sécurisé est mesuré par le TPM puis exécuté ;
  - Le chargeur sécurisé lance une application de confiance.

[3] Le TPM traite les requêtes séquentiellement et n'est pas conçu pour un environnement multithread

Le processus de mesure se déroule de la manière suivante : Soient A et B deux entités telles que A exécute B :
- A mesure B (un exécutable ou un autre fichier). Le résultat est un condensé de B ;
- Ce condensé est écrit dans un fichier log Stored Measurement Log (SML) stocké en mémoire.
- A écrit le condensé de B dans un registre PCR via l'opération d'extension des registres PCR ;
- Le contrôle est donné à B.

La mesure du poste client fait intervenir deux éléments cruciaux :
- Le fichier SML. Il s'agit d'un fichier de journalisation des mesures géré par l'application initiatrice de la mesure. Ce fichier contient le triplet suivant :
  - Le registre PCR utilisé pour la mesure ;
  - Le condensé SHA-1 ;
  - Le nom de l'élément mesuré.
- Les registres PCR. Leur fonction est de garantir l'intégrité du fichier SML. Le processus de vérification de l'intégrité du fichier SML se déroule en deux étapes :
  - Rejeu logiciel du processus d'extension des registres(Cf. formule 1) sur la base des mesures du fichier SML ;
  - Comparaison du résultat avec l'état courant des registres PCR du TPM.

  L'intégrité du fichier SML est valide si les résultats sont identiques. Cette opération doit être initiée par une application tierce de confiance. Le synoptique 3 décrit le déroulement du processus d'attestation d'une plateforme.

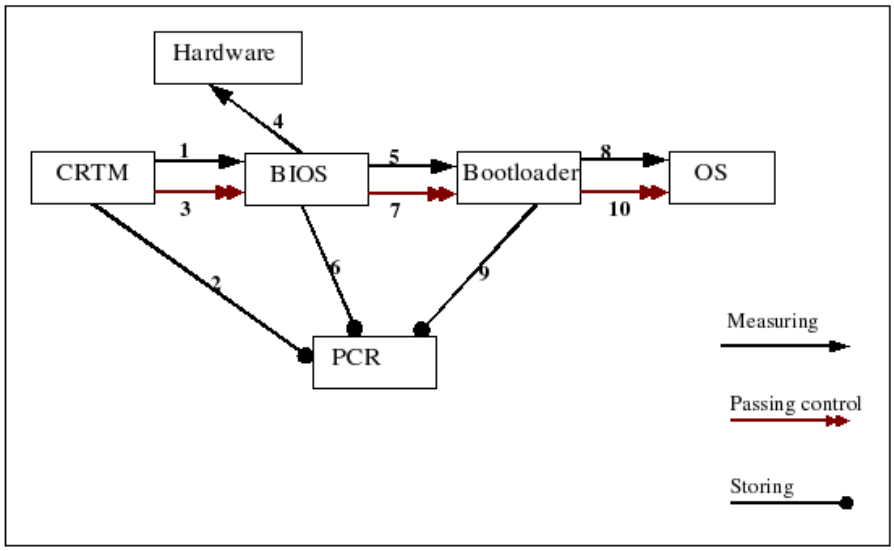


FIG. 3: Déroulement du processus de mesure

3.2 Attestation distante

L'attestation distante constitue le prolongement naturel du service de mesure du poste client. Elle offre l'opportunité à un tiers distant de vérifier l'état dans lequel se trouve une plate-

PCR	Mesure	Objets Mesurés
00	DB4C49FC7A337A8562080BC8BB030F69A114DA80	[]
00	D7BFDFDC4597548A092482A069BFDB1B2EAE60D1	[]
...		
00	6FF1CF841F11BDBAEC62464C0672D2BDFE7F5975	[]
01	DD85953BA1CA240ACB426D6001E719F41E41473D	[]
02	E79E468B1921B2293A80C5917EFA6A45C379E810	[START OPTION ROM SCAN]
01	7768CC191DA2BF9F5BF8C3B46F849CF038D66429	[CMOS]
01	017263855C5E8B20F2896A3135B8E4652AB1E708	[WAKE EVENT 0]
00	2907B0A74E2E025F863BDA3DD55A9ADA385DCF28	[Event Separator]
01	2907B0A74E2E025F863BDA3DD55A9ADA385DCF28	[Event Separator]
02	2907B0A74E2E025F863BDA3DD55A9ADA385DCF28	[Event Separator]
03	2907B0A74E2E025F863BDA3DD55A9ADA385DCF28	[Event Separator]
04	2907B0A74E2E025F863BDA3DD55A9ADA385DCF28	[Event Separator]
05	2907B0A74E2E025F863BDA3DD55A9ADA385DCF28	[Event Separator]
06	2907B0A74E2E025F863BDA3DD55A9ADA385DCF28	[Event Separator]
07	2907B0A74E2E025F863BDA3DD55A9ADA385DCF28	[Event Separator]
04	C1E25C3F6B0DC78D57296AA2870CA6F782CCF80F	[Calling INT 19h]
04	38F30A0A967FCF2BFEE1E3B2971DE540115048C8	[Returned INT 19h]
...		
10	629ffb8cc6fae6a445bcae9f3da003a833705cbd	boot_aggregate
10	bd32143f5b862d211ccf5769bad9fb90261f25fe	/bin/dash
10	326cc2a348fe4e6fd67d94b7323c3bf9ad465c01	/lib/ld-linux.so.2
10	7e91ec963274e20f00e96f30cf1c8af0e3604c6a	/lib/i686/cmov/libc.so.6
10	46ed06087da014f53482aa22fbdeacac48ce1a6d	/bin/mount
10	bb961020409bcd22d168f411aca1c5c30c5d7938	/lib/libblkid.so.1
10	4aab3bca412389e824c425ff0fbcf213ee58532f	/lib/libuuid.so.1
10	b24e34e8ec1c436144ace6486a2d3107251f75ef	/lib/libdevmapper.so.1.02
10	ba03dabb1439148f8a028ac7432b327cf59bada9	/lib/libselinux.so.1
10	9f3dee32465f1bcc26f43ece3311d2183495c47f	/lib/libsepol.so.1
10	6c53896b81f2149c5660a879ab0d0d68db8971fb	/lib/i686/cmov/libpthread.so.0
10	5c685289d970c026898fac8a15c249c1d876ed40	/lib/i686/cmov/libdl.so.2
10	288d33eb92ceda7529686a3d881385266e03ce3b	/bin/cat
10	60a85b2bac22bcc34483d11188b73a67c88dd239	/bin2/umount

TAB. 1: Exemple de fichiers de mesure (SML)

forme et d'en déduire si cet état est de confiance ou non. Lorsque la chaîne de confiance couvre le système d'exploitation et les applications, le demandeur d'une attestation peut alors conditionner la délivrance d'un contenu sensible à la valeur des registres PCR.
Le concept d'attestation distante repose sur l'existence d'une clef de base ou clef d'approbation « Endorsement Key (EK) » . Cette bi-clef RSA est programmée dans l'E^2PROM du TPM en phase de personnalisation du composant, elle est certifiée par une AC permettant d'associer à chaque TPM un certificat unique. En utilisant la partie privée de sa bi-clef, le TPM peut signer des éléments de sa chaîne de confiance, un tiers distant peut alors vérifier que ces signatures ont été générées via un TPM de confiance. Toutefois, afin de préserver la confidentialité de la clef EK, la signature est faite par une clef RSA d'attestation d'identité « Attestation Identity Keys (AIK) ». Une clef AIK est une clef générée par le TPM et signée avec sa clé EK.
La mise en place d'un service d'attestation distante pose l'épineux problème du respect de l'anonymat du possesseur du composant. La première version des spécifications des TPM $V1.1$ introduisait en effet un protocole faisant intervenir une AC privée. Ce protocole reposait sur la génération de clefs AIK à chaque instance du protocole. Il induisait intrinsèquement un problème d'anonymat et de disponibilité du TPM.

- **Anonymat**. La collusion possible entre les AC privées et les vérifieurs (*i.e.* les entités qui souhaitent attester la plateforme) remettait en cause l'anonymat en permettant notamment le traçage de la plate-forme attestée ;
- **Disponibilité**. La disponibilité du TPM était quant à elle compromise par plusieurs facteurs :
  - La certification des clefs AIKs qui requérait systématiquement une interaction avec l'AC ;
  - L'AC et le TPM pouvaient être submergés de requêtes compte tenu du grand nombre respectifs de TPM et de vérifieurs sur le marché.

La version 1.2 des spécifications du TCG permet de résoudre ces problèmes en introduisant un nouveau protocole : le Direct Anonymous Attestation (DAA) issu des publications de Jan Camenish d'IBM. Dans l'esprit, ce protocole est inspiré des techniques zero-knowledge. Le DAA atteste à une entité externe qu'un message a été signé par un TPM et certifié auprès d'une AC sans pour autant donner d'information sur l'identité ni du TPM ni de l'AC. A l'heure de la publication de cet article (Octobre 2008), l'implémentation de ce protocole demeurait encore très partielle.

3.3 Gestion des clefs

Les différents services utilisant les mécanismes de signature et de chiffrement reposent sur une hiérarchie de clefs (cf. figure 4). Chaque clef est protégée par une clef de niveau supérieur. La racine de cette hiérarchie est constituée par la clef SRK (Storage Root Key). Cette clef est stockée à l'intérieur du TPM, tandis que les autres clefs peuvent être stockées sur un support externe (disque dur par exemple). Dans ce dernier cas de figure, chaque clef est associée à une clef de stockage. La clef de stockage permet alors de chiffrer la clef lors de son stockage sur un support externe. En ce qui concerne la clef SRK, elle est générée par le TPM lors de la prise de possession du TPM en utilisant la commande `PM_TakeOwnership`.

Différents type de clefs sont spécifiés par le TCG, suivant l'opération qui en ait faite. Certaines sont dédiées au chiffrement (*binding keys*). Elles ont alors comme propriété d'être transférables d'un composant TPM à un autre. À ce titre, le contenu chiffré peut être déchiffré par une autre plate-forme. Des clefs peuvent également servir pour sceller des données (*sealing keys*). Dans ce cas, elles ont comme propriété d'être non-transférables. Chaque contenu scellé est associé à un unique TPM. Enfin, des clefs sont utilisées pour signer certains objets internes au TPM. Typiquement, les clefs AIK-qui ne sont pas transférables-peuvent signer une liste

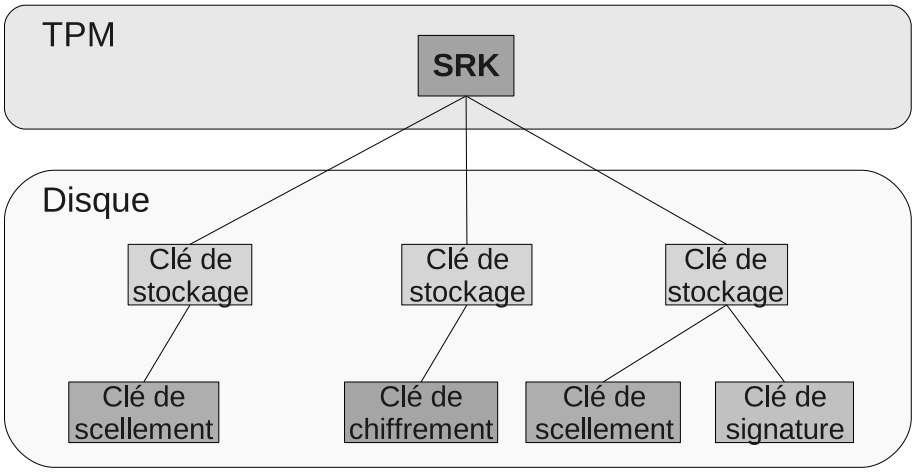


FIG. 4: Hiérachie des clefs

de registres PCR. Ce dernier mécanisme sert principalement pour le service d'attestation distante.

3.4 Gestion des certificats

L'utilisation des certificats fait partie intégrante du mécanisme d'attestation distante. L'objectif est de prouver à une autre entité la conformité du TPM local vis-à-vis des spécifications du TCG. Cela passe notamment par une chaîne de certification en amont de la distribution du composant et du matériel l'exploitant. Les certificats présentés ici respectent la norme X.509 avec la syntaxe ASN.1.
Tout d'abord, une autorité de certification (par exemple VeriSign) certifie le fabriquant de la puce TPM (par exemple Infineon ou Atmel). Cette autorité fournit alors un certificat appelé *conformance credential*, qui fait office de certificat racine et sert à prouver la conformité de la fabrication du TPM. Le fabriquant de la puce TPM certifie à son tour le fabriquant de la carte mère qui intègre la puce dessus. Le certificat qui en résulte est appelé *Platform credential*. Il prouve que le processus d'intégration du composant est réalisé en respect des normes. Ce fabriquant initialise alors le module TPM avec une clef unique appelée *endorsement key* (EK). Il en profite pour signer cette clef et créer un certificat appelé *endorsement credential*. Il est normalement livré avec la carte mère, soit directement inclus dans la puce TPM, soit inséré dans le CD-ROM qui l'accompagne. L'*endorsement credential* sert à attester à un tiers que la puce est physique et qu'elle respecte le processus d'intégration et de certification.
Le protocole d'attestation fait quant à lui intervenir un dernier certificat, appelé *AIK credential*. Il est envoyé par le *Privacy CA* lors du mécanisme d'authentification et sert lors de procédures de signatures.

4 Retour d'expériences

Cette partie présente les résultats des tests de conformité et d'efficacité réalisés dans le cadre du marché DGA Postes Clients et Interconnexions Multiniveaux (PCIM).

4.1 Stratégie de Tests

La stratégie de tests du composant et de sa pile logicielle s'est inspirée du schéma d'évaluation/certification CSPN de la DCSSI. Ce schéma permet de tester des produits de sécurité en temps contraint selon une méthodologie privilégiant l'expertise technique à la complétude méthodologique. Les quatre grandes phases de la stratégie de tests sont :
- Description des fonctionnalités, de l'environnement d'utilisation et de sécurité ;
- Analyse de la conformité permettant de vérifier la bonne implémentation des mécanismes ;
- Analyse de la résistance des fonctions et des mécanismes permettant de vérifier l'efficacité sécuritaire des fonctions ;
- Analyse des vulnérabilités permettant de décrire les vulnérabilités connues, d'identifier les vulnérabilités inconnues *a priori*.

La réalisation d'attaques matérielles ne figurait pas dans le périmètre de nos tests. Les résultats de l'implémentation d'attaques par canaux cachés (SPA, DPA, HO-DPA,...), par injections de fautes ou autres attaques intrusives sur des composants TPM ne sont pas légion dans la littérature publique. La majorité des constructeurs assurent néanmoins que leurs puces respectives implémentent des contre-mesures à ces attaques. On citera notamment le tutorial « Deep Inside TPM » présentant les protections implémentées par le constructeur INFINEON dans sa gamme de composant TPM 1. 2 [11].

4.2 Synthèse de l'analyse des applications

Portée des tests. Les expérimentations ont principalement concerné les services suivants :
- La mesure du poste client ;
- La pile logicielle TSS.

Les tests sur le service de mesure du poste client ont été effectués sur les applications suivantes :
- Trusted Grub ;
- Integrity Measurement Architecture (IMA) ;
- eCryptfs ;
- XEN.

L'objectif des tests sur le service de mesure du poste client est de vérifier la conformité des applications relativement à leurs spécifications. La conformité est testée en modifiant des éléments critiques du poste client et en observant :
- Le contenu du fichier SML et des registres PCR ;
- La cohérence des mesures réalisées avec les valeurs des registres PCR.

Les tests sur la pile logicielle TSS ont été effectués sur les applications suivantes :
- Trousers ;
- TPM-TOOLS.

Les tests sur la pile TSS ont permis d'estimer -via un processus de test par échantillonnage- la conformité de la pile TSS par rapport à ses spécification et de mettre en œuvre une attaque de la pile.

Organisation des tests. Les tests se sont déroulés de la manière suivante :
- *Tests sur l'association Trusted Grub et IMA.* Cette association implémente le service de mesure du poste client. Trusted Grub constitue une extension de Grub réalisant les mesures des éléments logiciels intervenant dans la phase pré boot de l'OS. Une fois l'OS opérationnel,

IMA assure le relais en mesurant les binaires chargés en mémoire. Le fonctionnement de ces applications est détaillé dans les chapitres ci-après.
L'objectif des tests était de vérifier la conformité du processus de mesure. La mise en œuvre a été réalisée en modifiant certains éléments logiciels afin de vérifier la répercussion de ces changements sur les mesures. Les éléments logiciels critiques modifiés potentiellement exploitables par un attaquant sont :

- La séquence de boot paramétrable dans le BIOS ;
- La zone d'amorce du disque dur (MBR) ;
- Le stage 1.5 de Trusted Grub ;
- Le fichier de configuration de Grub ;
- Les fichiers chargés par Trusted Grub pour le démarrage du système d'exploitation (le noyau, l'image du système minimal initial : initrd) ;
- Les fichiers exécutables et les librairies partagées.

A chaque modification, le poste client est redémarré. L'évolution du fichier SML et les valeurs registres PCR sont ensuite observés. La cohérence des mesures avec les valeurs des registres PCR est réalisée en itérant le processus d'extensions des registres. Ainsi, il est possible de détecter :

- La non-prise en compte d'une modification du poste client ;
- La non-extension des registres PCR par rapport à une mesure faite ;
- La mesure d'un élément non enregistré dans le fichier SML.

– *Tests sur eCryptfs.* eCryptfs est un système de chiffrement de fichiers fonctionnant sous Linux. Ce système de fichier est annoncé comme compatible du TPM. La mise en œuvre des tests a consisté à créer un système de fichiers chiffrés utilisant le scellement. La conformité a été testée via la vérification de l'impossibilité de monter le système de fichier si l'état des registres PCR est corrompu.
– *Tests sur XEN.* XEN a développé un hyperviseur capable de mettre en place des TPMs virtuels. Les tests sur XEN se sont concentrés sur la chaine de confiance qui peut exister entre la machine hôte et les machines virtuelles. La conformité a été testée via la modification de certains éléments mesurés de l'hôte et la vérification des répercutions sur les TPMS virtuels.
– *Tests sur la combinaison de Trousers et TPM-TOOLS.* Trousers constitue l'implémentation de la pile TSS sous Linux. TPM-TOOLS est une boite à outils permettant d'utiliser le TPM. Les tests se sont concentrés sur la conformité de la fonction de scellement et descellement et sur la mise en œuvre d'une attaque de type API hooking sur la pile.

Trusted Grub. Trusted Grub est un patch à appliquer sur le chargeur de démarrage de Grub. Il mesure les éléments suivants lors du démarrage du poste :

– Le stage 1.5 de Grub[4] ;
– Le stage 2 de Grub ;
– Le fichier de configuration de Grub ;
– Le noyau Linux ;
– L'image du système minimal (`initrd`).

Les tests effectués se sont concentrés sur l'analyse de la conformité des mesures et ont consisté à :

– Modifier un élément ;
– Redémarrer la plateforme ;
– Consulter le fichier SML afin de vérifier des changements dans les mesures.

[4] On notera que le stage 1 de Grub est mesuré par le BIOS.

À l'issu des tests, nous avons constaté la conformité de la mise à jour du fichier SML. Néanmoins, l'étape de reproduction du processus d'extension des registres PCR a mis en exergue une non conformité fonctionnelle. Une analyse du code nous a permis de constater que la fonction d'extension a été omise.

Integrity Measurement Architecture. IMA est un module noyau permettant au système d'exploitation de mesurer l'intégrité des binaires chargés en mémoire : pilotes, librairies et exécutables. IMA est développé par le groupe de recherche d'IBM sur le TCG([6]). Il se présente sous la forme d'un patch du noyau Linux. Le fonctionnement d'IMA est transparent pour l'utilisateur. Lors du chargement du binaire en mémoire, IMA le mesure. Si le binaire a déjà été utilisé, IMA vérifie le motif d'intégrité du binaire avec celui du fichier SML. Quelque soit le résultat de ce test, IMA n'empêche pas l'utilisation des binaires. En revanche, il incrémente des compteurs reflétant les résultats de ces mesures :
- Nombre de fichiers mesurés ;
- Nombre de changement d'intégrité ;
- Nombre de violations ;
- ...

De plus, IMA offre également aux applications la possibilité de mesurer un fichier en lecture seule.
Les tests se sont concentrés sur la conformité du produit. Nous avons étudié les mesures de binaires et l'interface de mesure des fichiers en lecture seule. Le produit s'est montré conforme. IMA reste néanmoins plus proche d'un prototype expérimental que d'un produit finalisé. Une application tierce interprétant les mesures d'IMA afin de réagir aux changements d'intégrité est notamment très attendue.
L'association d'IMA avec Trusted Grub est un exemple d'implémentation du service de mesure du poste client. Malheureusement, le manque de synergie entre les spécifications des deux produits affaiblit la chaîne de confiance. En effet, IMA utilise les premiers registres PCR pour initialiser ses mesures alors que les paramètres par défaut de Trusted Grub prévoient une mesure du noyau Linux (et donc d'IMA) sur les registres suivants.
A la lumière des tests réalisés et des informations disponibles sur les sites publics, il n'existe pas d'application tierce totalement fonctionnelle implémentant un système de vérification des mesures. Les BIOS compatibles TPM n'intègrent pas encore de tels systèmes de vérification (ils réalisent uniquement des mesures). L'implémentation conjointe d'un système de réalisation de mesures, de vérification de mesures et décision conditionnelle (poursuite de l'exécution ou non) nous semble aujourd'hui une tâche complexe notamment au niveau de l' implémentation (stockage du fichier de référence, mise à jour du fichier de référence, intégrité du fichier de référence,...). Par transitivité, cette propriété du BIOS doit être applicable au bootloader et au système d'exploitation. La deuxième solution pour vérifier l'intégrité de la chaîne d'attestation serait d'utiliser un processus de scellement à tous les niveaux (BIOS, bootloader, 0S,...). Malheureusement, nos tests ont montré que ces fonctionnalités n'étaient pas encore implémentées dans le BIOS, trusted Grub et l'OS.

Trousers. Trousers est un projet Linux visant à développer une pile logicielle TSS de source libre. Elle permet à des applications comme Tpm-tools d'utiliser le TPM. À l'heure actuelle, Trousers est compatible intégralement avec les spécifications 1.1 du TCG mais partiellement avec la version 1.2. Néanmoins le projet en est à un stade très avancé et la pile est aujourd'hui fonctionnellement quasi complète. Par ailleurs, une analyse en temps contraint de la pile a été effectuée selon une métrique d'analyse d'API cryptographique comprenant des critères tels que :

- L'indépendance vis à vis des algorithmes et des applications ;
- L'indépendance vis à vis du module cryptographiques ;
- Le degré d'expertise cryptographiques (prises en compte des attaques de la littérature, conception sécurisée) ;
- Les services auxiliaires (cycle de vie des clefs, audit des requêtes, authentification du module, ...) ;
- Le partage de charge, gestion de l'asynchronisme ;
- Le contrôle de flux ;
- ...

Les tests effectués montrent que la version actuelle de la TSS a plutôt l'apparence d'une API de type « prototype » que d'une API finalisée. Les quelques exemples identifiés nous laissent penser que les principaux mécanismes de sécurité ne sont pas encore implémentés. Typiquement, nous n'avons que peu d'informations sur la capacité de la pile à résister aux principales attaques de la littérature publique.

A titre illustratif, nous avons développé une attaque de type API Hooking de la pile TSS. L'attaque permet de récupérer le mot de passe de la clef SRK et l'intégralité de la clef AES utilisées pour chiffrer les données scellées. La technique d'API Hooking ne modifie pas l'intégrité de l'application de scellement et de la pile TSS. La technique demande uniquement les droits de l'utilisateur.

TPM-TOOLS. TPM-TOOLS constitue une boîte à outils permettant d'utiliser le TPM. Son fonctionnement requiert l'utilisation de Trousers. Les services proposés par TPM-TOOLS :

- Scellement de données ;
- Activation du TPM ;
- Mise à zéro du TPM ;
- Obtention de la partie publique de l'EK ;
- Prise de possession du TPM (génération d'une nouvelle clef SRK et révocation de l'ancienne).

Nos analyses se sont concentrées sur une étude de la conformité fonctionnelle de l'outil.

4.3 Association du TPM avec eCryptfs

eCryptfs est un système de chiffrement de fichiers fonctionnant sous Linux. Il se place au dessus d'un système de fichiers traditionnel et utilise une clef de chiffrement symétrique unique pour chaque fichier. La confidentialité de cette clef repose typiquement sur un chiffrement RSA via le TPM. eCryptfs utilise la fonction de scellement du TPM (`Sealed Storage`) pour stocker de manière sécurisée les clefs symétriques. Cela permet de faire dépendre le déchiffrement des données de l'intégrité de la machine, et plus particulièrement de l'état des PCR. Actuellement, le support du TPM dans eCryptfs n'est pas mature. En effet, plusieurs problèmes d'implémentation subsistent. Par exemple, l'authentification avec la puce TPM n'est pas effectuée de manière respectueuse des spécifications du TCG. Dans certaines situations, une modification du code source d'eCryptfs est alors nécessaire. De plus, le choix des PCR à utiliser lors du scellement doit être effectué par l'utilisateur. Celui-ci doit donc avoir une très bonne connaissance du fonctionnement du TPM et du principe de chaîne de confiance sous-jacent. Enfin, certains bugs empêchent la lecture de données chiffrées juste après le montage d'une partition. De ce fait, il n'apparaît pas envisageable d'utiliser eCryptfs avec le support du TPM dans un environnement de production à l'heure actuelle.

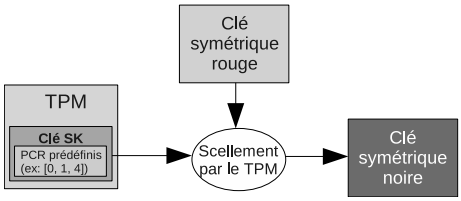


FIG. 5: Scellement de la clef symétrique

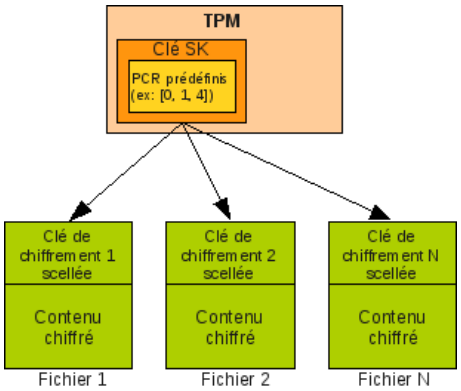


FIG. 6: Association eCryptfs/TPM

4.4 Virtualisation TPM

Présentation de la virtualisation. La virtualisation TPM a pour objectif de fournir les services du composant matériel aux machines virtuelles. À l'heure actuelle, deux solutions liées au concept de virtualisation sont présentes sur le marché :
- Une solution logicielle reposant sur la solution de virtualisation XEN ;
- Une solution d'IBM reposant sur un support matériel mettant à disposition un composant TPM physique virtualisable.

Solution reposant sur XEN. La solution reposant sur XEN consiste à émuler logicielle-ment un composant TPM au niveau de l'hyperviseur. Il s'agit d'une implémentation logicielle où l'utilisation du TPM physique est très limitée.
Les avantages et inconvénients d'une telle solution sont liés du fait qu'une instance virtuelle, logicielle, de TPM (vTPM) est clairement différenciée d'une implémentation matérielle et les propriétés de sécurité sont différentes :
- Les hiérarchies de clefs sont indépendantes : chaque vTPM possède sa propre hiérarchie de clefs, sans lien avec la hiérarchie des clefs du TPM physique (notamment les *storage root key* et *endorsement key*) ;
- La génération des clefs d'un vTPM est exécutée en logiciel par le processus et ne fait pas intervenir le TPM matériel ;
- Les fonctions des vTPM sont indépendantes des fonctions du TPM : un appel à un vTPM ne se matérialise pas par un appel au TPM.

Nos tests sur ce produit se sont cantonnés à :

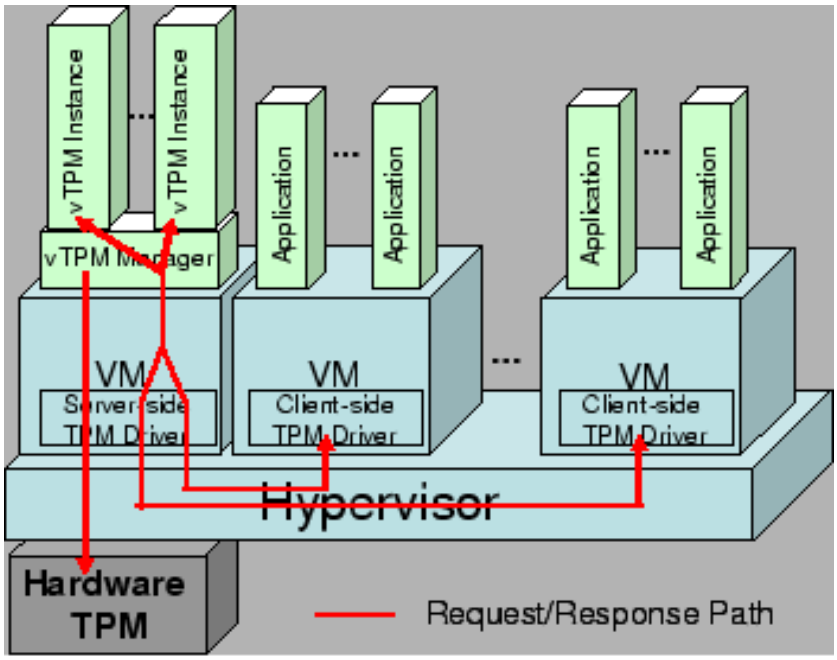


FIG. 7: Virtualisation de TPM – solution de XEN

- Vérifier la possibilité d'utiliser le vTPM dans une machine virtuelle ;
- Vérifier la continuité de la chaîne de confiance (propagation des mesures).

Les expérimentations ont montré que :

- La solution de XEN correspond fonctionnellement à ses spécifications : les machines virtuelles utilisent bien les services des vTPM ;
- La continuité de la chaîne de confiance n'est pas réalisée puisque l'initialisation d'un vTPM ne fait pas intervenir l'état de la machine hôte. Selon nous, une solution aurait été d'initialiser les registres PCR du vTPM avec une mesure de XEN et des registres PCR du TPM matériel.

Solution d'IBM. L'initiative d'IBM part de la constatation que les TPM n'ont pas été spécifiés pour être employés par plusieurs systèmes à la fois. Par conséquent, IBM a étendu les spécifications du TPM permettant un usage multi-système simultané. Cette solution fait l'objet d'une extension au standard TPM 1.2, définissant de nouvelles commandes dédiées à la virtualisation d'un TPM. Le support matériel (IBM PCI-X Cryptographic Processor, carte PCIXCC) met à disposition un composant TPM physique aux machines virtuelles. Les TPM virtualisés sont indépendants les uns des autres, ne partagent pas d'informations et ne communiquent pas. Chaque machine virtuelle a accès uniquement à son TPM virtuel à travers le proxy et est cloisonnée des autres TPM virtuels. La seule possibilité de partage d'informations entre instances est d'utiliser les fonctions de sauvegarde et restauration d'instance pour créer

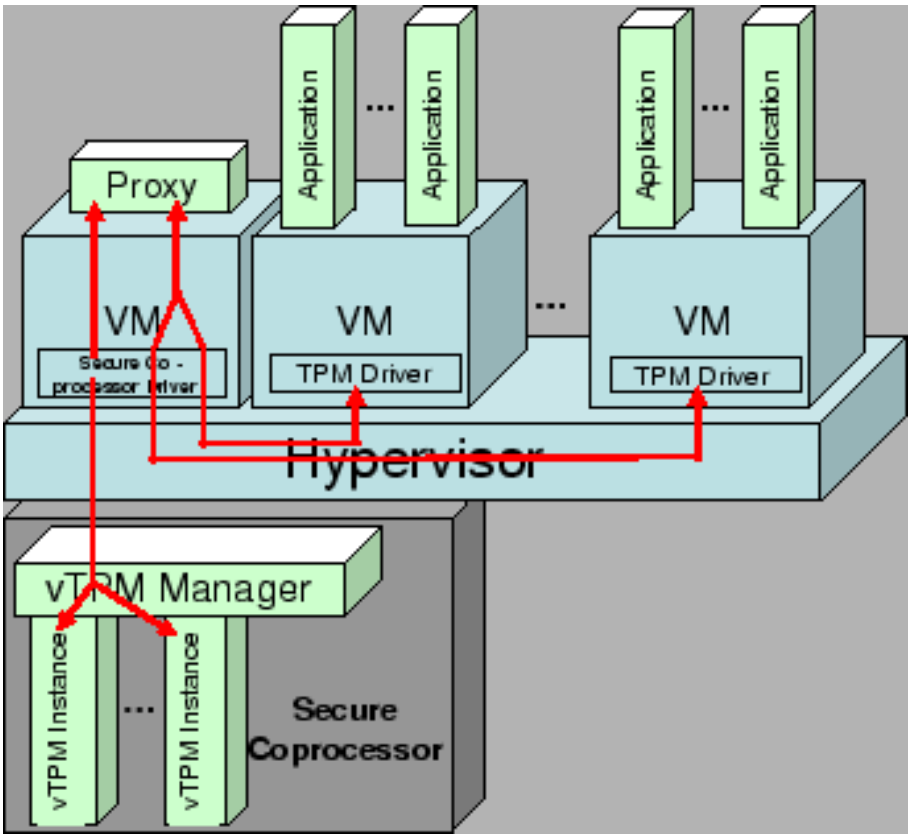


FIG. 8: Virtualisation de TPM : solution d'IBM

des clones d'un même TPM virtuel. Il faut toutefois remarquer que la gestion de l'isolation entre les vTPM est en partie logicielle : c'est au proxy, qui concentre toutes les requêtes des machines virtuelles, d'assurer l'aiguillage des commandes en fonction des associations machine virtuelle / vTPM.
De plus, le système prévoit qu'une machine virtuelle puisse virtualiser elle même un composant TPM virtuel. Dans ce cas, le TPM virtuel associé à cette machine virtuelle est lui-même virtualisé pour les machines virtuelles filles (ceci se traduit en pratique par le fait qu'une commande transite deux fois par le proxy avant d'atteindre la carte). Ceci conduit à une arborescence de TPM virtuels, qui peuvent s'appuyer soit sur le TPM physique soit sur un autre TPM virtuel. Cette fonctionnalité traduit le caractère modulaire et avancé de la solution.

5 Conclusion

La technologie TPM constitue une avancée indéniable pour la sécurisation des architectures de postes clients. Elle offre nativement des services cryptographiques implémentés matériellement à moindre coût et permet la spécification de services de sécurité haut-niveaux tels que la mesure du poste client ou l'attestation distante. Nos expérimentations sur le composant TPM,

sa pile logicielle et les différentes applications utilisatrices ont montré que certains problèmes de conformité fonctionnelle et d'efficacité subsistent. On citera notamment :
- Le manque de maturité, l'absence de prise en compte de la sécurité dans la pile TSS ;
- Le problème de compatibilité matérielle avec certains composants du marché (BROADCOM) ;
- Le problème de l'implémentation du service d'authentification d'eCryptfs avec la puce TPM.

Toutefois, ces problèmes sont à relativiser compte tenu de la jeunesse de cette technologie et des nombreux développements en cours de réalisation. Par ailleurs, notre confiance dans cette technologie est ternie par la relative opacité des spécifications du TCG. En effet, ces dernières sont très volumineuses mais malheureusement peu explicites et trop adaptables par les constructeurs sur certains aspects (mesure du boot, protocole d'activation, la mesure du poste client et l'attestation distante, ...). Elles s'apparentent plus à des documents de conception détaillés destinés aux développeurs. Il manque clairement un niveau de spécification formelle ou au moins pseudo-formelle des principaux protocoles et services cryptographiques. Il conviendra donc de suivre attentivement les futures évolutions de cette technologie, notamment en terme d'évaluation de sécurité, afin d'en garder une certaine maîtrise garantissant la confiance dans son utilisation.

Références

1. TPM Main Part 1 Design Principles Specification Version 1.2 Revision 94 29 March 2006
2. TPM Main Part 2 TPM Structures Specification Version 1.2 Revision 94 29 March 2006
3. TPM Main Part 3 Commands Specification Version 1.2 Level 2 Revision 94 29 March 2006
4. TCG Specification Architecture Overview Specification Revision 1.3 28 March 2007
5. TCG Software Stack (TSS) Specfication Version 1.2 Level 1 Errata A Part1 : Commands and Structures 7 March 2007
6. Reiner Sailer and Xiaolan Zhang and Trent Jaeger and Leendert van Doorn, Design and Implementation of a TCG-based Integrity Measurement Architecture
7. Stefant Berger, Ramon Caceres, Kenneth A. Goldman, Ronald Perez, Reiner Sailer, Leendert Van Doort, vTPM : Virtualizing the Trusted Platform Module
8. Virtual Trusted Platform Module `http://domino.research.ibm.com/comm/research\_projects.nsf/pages/ssd\_vtpm.index.html`
9. TPM Main Part 3 IBM Commands Specification Version 1.2 Revision 10 25 April 2005 `http://domino.research.ibm.com/comm/research\_projects.nsf/pages/ssd\_vtpm.index.html/\$FILE/mainP3IBMCommandsrev10.pdf`
10. Bernhard Kauer, `http://www.cs.dartmouth.edu/~pkilab/sparks/`
11. Hans brandl, Trust 2008 conference, Villach-Austria

Efficient Techniques for Securing Off-Chip Memory

Juan Garay, Vladimir Kolesnikov, and Rae McLellan

Bell Labs, Murray Hill, NJ 07974,USA
{garay|kolesnikov|rae}@research.bell-labs.com

Abstract We consider a system architecture where a hardware-secured processor uses memory controlled by an adversary. We present a new, more efficient approach to encrypting and authenticating memory and discuss the associated trade-offs, while paying special attention to minimizing hardware costs and the reduction of DRAM latency.

Instrumental in our approach is *ShMAC* (Shallow MAC), a novel fixed input length message authentication code that allows to perform most of the computation *prior* to the availability of the message. Specifically, ShMAC's message-dependent computation is much faster and smaller in hardware than the evaluation of a PRP, and can be implemented by a small *shallow* circuit, while its precomputation consists of one PRP evaluation.

We describe our architecture in detail, and show the clear advantages of MAC precomputation in our system.

Keywords : MAC, precomputation, System on a Chip, secure processor

1 Introduction

With the highly publicized attacks on consumer computer products, such as the iPhone [10] and Xbox, security of computing has become a topic of widespread commercial interest. Broadly speaking, security of computing can be divided into two main areas — hardware and software security. Software security is concerned with integrity of the software and whether it can be suborned to yield control or sensitive information to an outside attacker. Hardware security, on the other hand, assumes that the adversary has full physical access to the device and may use oscilloscopes and logic analyzers to observe and compromise the computing system. This paper focuses on ways to efficiently provide hardware security.

Recent advances in VLSI have provided strongly tamper-resistant hardware computing platforms by integrating complete Systems on a Chip, through SoC technology. It is considered infeasible to all but government-scale attackers to perform meaningful analysis of the internals of production SoC. An ideal solution would be to integrate the entire computing function onto a single chip, with the computing function's data and program remaining entirely on-chip, and encrypting all off-chip communication. However, in most practical scenarios, the amount of on-chip memory is insufficient to hold the entire computing function's program and/or data, due to prohibitive costs of such SoC. In this paper we consider the question of how to encrypt off-chip DRAM (Dynamic Random Access Memory) transactions with minimal performance degradation and cost increase. In a computing system, off-chip DRAM transactions

occur much more frequently than network messages and have much more stringent latency requirements. Since processor performance is so tightly dependent on off-chip memory latency, speeding up the encryption/authentication process is of primary importance.

As it turns out, for many on-chip bus protocols the address is available early in the bus transaction between the processor and memory controller, while the wider data follows later and is composed of multiple transfers of narrower sub-units. This common technique of serializing data transfers in on-chip buses is an engineering trade-off between performance and the number of wires required for a wider bus. Therefore, an encryption/authentication algorithm which can postpone data-dependent computation, can start earlier in the memory transaction and potentially reduce the system performance impact of an encrypted memory system. This paper describes an efficient way to encrypt off-chip memory transactions and provide data authentication that takes advantage of the early arrival of the memory address.

Instrumental in our approach is a fixed input length Message Authentication Code (MAC) construction which allows the bulk of the MAC computation to be performed *before* the message m is available. The computation dependent on m is the evaluation of (a new variant of) an ϵ-*differentially uniform* function [19,18] (cf. Section 2) and an XOR, which is much simpler and faster than a typical MAC implementation via a block cipher. In envisioned instantiations, the precomputation of the MAC is a PRP (e.g., a full 10-round AES) evaluation, and the remaining computation (dependent on m) is an evaluation of 2- or 4-round AES.

In addition to presenting our secure DRAM architecture, we discuss at length security-efficiency trade-offs, and underlying design choices.

Threat model. In this work we assume a powerful attacker who is able to act as Man-in-the-Middle on the channel between the CPU and DRAM. As discussed above, we assume that SoC internals are unavailable to the attacker. In our system, we guarantee protection against spoofing (creating arbitrary memory blocks) and relocation (moving valid memory blocks between memory locations) attacks on the DRAM. As an efficiency trade-off, we do not guarantee full protection against replay attacks (serving stale data from DRAM), but we discuss ways of mitigating this threat.

1.1 Related Work on Secure Memory

There is a vast amount of work on securing memory. One direction uses smart cards or other separate adjunct chips such as the Trusted Platform Module, or TPM [23] in secure systems design. These methods are usually limited, in the sense that they do not protect intellectual property contained in the software, but only secure execution of small parts of it by executing it on the smart card/TPM . These systems (especially TPM-based) are generally used to authenticate the execution of software running on an attached host processor, but are unable to withstand a physical hardware attack nor provide computational privacy for the overall system. An interesting use of a smart card processor was proposed in the XμP system [2]. XμP allows the ROM-less smart card to execute signed code, using the terminal as a (cheap) storage. The authors describe ways of securing the computation, including a public- and symmetric key-based authentication of the executing code. The symmetric-key case resembles our setting at a high-level; however XμP is not as severely restricted, uses expensive hash functions, and does not approach the issue of MAC precomputation.

Another encrypting memory system is XOM [16], which provides architectural support for software licensing and allows code to be authenticated and run even under untrusted operating systems. The XOM system require significant modification of a processor's instruction cache, the addition of special instructions, as well as operating system support to function properly.

The system we propose in this paper is independent of the instruction set and supports any processor architecture with little or no operating system support. Our goal is not to provide licensing support so much as to provide reliable computation privacy.

Closer to our setting, securing memory in a SoC system has been recently announced by IBM [11] and considered academically in several projects such as AEGIS [22,21], Crypto-Page [6], TEC-Tree [7]. These systems validate their encrypted memory state by building and continuously maintaining a hash tree of the entire memory space for protection from replay attacks. While it has been noted that caching these operations greatly reduces the performance impact [9], to do so requires significant on-chip resources to build such a cache. If the processor's own cache system is to used to hold both data and the Merkle tree [17] of MACs, then there's the system level question of how to handle non-processor originating data such as DMA or IO traffic. Also, modifying a processor's cache system is not always possible, as for example the hard coded PowerPC processor embedded in Xilinx FPGAs.

Other systems such as PE-ICE [8] or TEC-Tree [7], forgo Merkle trees but require significant on-chip storage for nonce values updated on each memory write operation. While the amount of on-chip storage can be as little as a byte for each encrypted off-chip storage block, this method doesn't readily scale to to support the desired gigabytes of off-chip DRAM.

In face of these overheads we have chosen to forgo replay attack protection altogether and rely merely on changing the encryption keys at reasonably frequent intervals. In regards to encryption and MAC generation we have focused on efficiency and mimimal additional on-chip resources. We propose a system where authenticating a memory access takes slightly more than a a PRP evaluation, and is effectively further reduced by MAC precomputation. We discuss security/efficiency trade-offs in more detail in Section 4.

Note, fixed input length (FIL) MAC schemes that allow precomputation have been considered in the literature. Further, tools and methods similar to the ones we use have previously been analyzed and employed. We postpone the discussion of work related to FIL MAC to Section 5.

1.2 Organization of the Paper

In Section 2 we introduce the necessary notation, definitions and building blocks that we will be using.

In the cryptographic core of this work, Section 3, we first discuss the intuition of, and then formally present our MAC construction — ShMAC, together with an evaluation of its performance, and instantiation considerations. We postpone proofs to the full version of the paper.

In Section 4, we present in detail the system aspects of our secure memory architecture. In particular, we discuss security objectives, restrictions and assumption of our system, and its use of ShMAC.

The level of technical detail of Sections 2 and 3 is necessary to present ShMAC; however, Section 4 is readable independently of Sections 2 and 3. Further, interested reader can additionally refer to the full version for proofs and other technical details.

2 Preliminaries

Throughout the paper, we let k denote the security parameter, and we usually denote keys by $\ell \in \{0,1\}^k$. We denote a pseudo-random permutation generator by PRPG, and pseudo-random permutation by PRP. The constructions of this work assume the existence of PRPGs.

2.1 ϵ-Differential Uniformity and Properties of AES Rounds

A main building block for our MAC construction is *Strongly Differentially Uniform* (SDU) functions, introduced in Section 3.2. SDU function family is a stronger version of a *Differentially Uniform* (DU) family, which is widely used in block cipher design and which we present below as background.

For our particular application we will use the sub-class of *keyed* ϵ-DU *permutations*, due to their efficiency. Therefore, for simplicity, we do not discuss here ϵ-DU functions in their full generality. However, we note that unkeyed permutations or functions [19] could also be used in our constructions, and our analysis (appropriately modified for indices, etc.) equally applies. Let $\mathcal{G} : \{0,1\}^k \times \{0,1\}^k \rightarrow \{0,1\}^k$ be a keyed permutation family. Let $\Delta x, \Delta y \in \{0,1\}^k$ be fixed and let $X \in \{0,1\}^k$ be a uniformly distributed random variable. Let $G_l \in \mathcal{G}$ be a member of $\mathcal{G}$. (In the sequel, we may sometimes omit index ℓ and write $G_l \in \mathcal{G}$, when ℓ is clear from the context or where it does not play a role.) The *differential probability* $DP(\Delta x, \Delta y, \ell)$ is defined as

$$DP(\Delta x, \Delta y, \ell) = \Pr_X[G_\ell(X) \oplus G_\ell(X \oplus \Delta x) = \Delta y]. \quad (1)$$

Here Δx and Δy are viewed as input/output differences. The *expected* differential probability $EDP(\Delta x, \Delta y)$ is the expectation of $DP(\Delta x, \Delta y, \ell)$, over all keys ℓ. We are interested in the *maximum EDP* (*MEDP*) :

$$MEDP(\mathcal{G}) = max_{\Delta x, \Delta y \in \{0,1\}^k \setminus 0} EDP(\Delta x, \Delta y). \quad (2)$$

Informally, a small *MEDP* value corresponds to good bit mixing by $\mathcal{G}$ — indeed, small *MEDP* means that any change in the (randomly chosen) input of the cipher results in an unpredictable output. However, small *MEDP* does not necessarily imply "security under multiple queries," since the *MEDP* experiment is defined over all keys ℓ.

Definition 1. *We say that a permutation family $\mathcal{G}$ as defined above is* ϵ-Differentially Uniform *(ϵ-DU), if $MEDP(\mathcal{G}) \leq \epsilon$.*

It is well known [13,5] that the *MEDP* of two-round AES (AES2) is at most $1.6 \cdot 2^{-28}$, and the *MEDP* of four-round AES (AES4) is at most $1.8 \cdot 2^{-110}$. Thus, AES2 is a $1.6 \cdot 2^{-28}$-DU permutation, and AES4 is a $1.8 \cdot 2^{-110}$-DU permutation.

3 ShMAC : MAC with Precomputation

In this section we present Shallow MAC (ShMAC), a MAC scheme which takes advantage of precomputation. The required precomputation essentially consists of one PRP evaluation, while the message-dependent portion is a small shallow circuit, which can be evaluated in a fraction of the time required for a PRP evaluation. (As an added bonus, in our envisioned instantiation, precomputation can share hardware gates with the rest of MAC computation. This is a critical advantage in cases where chip area is restricted, as it is in FPGAs.)

Recall from Section 1 that we require a low-latency MAC scheme, simultaneously "cheap" to implement in hardware, and faster than a PRP (such as AES) or a hash evaluation. This requirement precludes many standard MAC solutions, such as AES-based, which require availability of the message at the very beginning of the computation. (To be concrete about the latency of these approaches, recall that AES requires the sequential evaluation of at least

10 rounds[1]. Further, many (but not all) Universal Hash Function (UHF)-based constructions require expensive group arithmetic and additional hardware, and thus are unacceptable in this setting. See Section 5 for more details.)

However, as also discussed in Section 1, in many systems, the address of the memory transaction arrives before the data, and thus the hardware MAC unit is idle waiting for the data. We explore the possibility of using these idle cycles to perform *precomputation* to speed up the MAC generation.

3.1 Intuition of ShMAC

Note that ϵ-DU permutation family $\mathcal{G}$ (e.g., 2 or 4-round AES), an object with much weaker security properties than a PRPG, can be the core of a MAC, with appropriate pre- and postcomputation.

Indeed, $G \in_R \mathcal{G}$ provides good bit mixing, but only on random inputs. We satisfy this by using (nonce-based new and secret) precomputed randomness to mask the data prior to each application of G. This masking of the inputs additionally prevents adversary $\mathcal{A}$ from collecting any information on (the key of) G, even if $\mathcal{A}$ sees MAC evaluated on messages of his choice (i.e., queries the MAC oracle adaptively).

Note, even though $\mathcal{A}$ has no knowledge of the random mask $mask_r$ derived from a nonce r, he can attempt a forgery using the same r (and thus the same $mask_r$). Output unpredictability guarantees of DU functions are too weak to protect against this attack, since in our scenario $\mathcal{A}$ knows $G(d \oplus mask_r)$ [2]. We strengthen the notion of DU to preserve its guarantees even after one query to G – see Definition 2 below. In terms of implementation, it turns out that masking the output of G with fixed secret randomness (which can be viewed as part of G's key) is sufficient to satisfy the stronger requirements, and results in secure MAC.

Details of the above are presented below in Sections 3.2 and 3.3.

3.2 ϵ-Strongly Differentially Uniform Functions

In this section we introduce the new notion of *Strong Differential Uniformity* (SDU), discuss its relationship with DU, and present an efficient construction. The new notion is a natural building block in MAC constructions, including ours, and may have applications in other areas. For simplicity, we give an asymptotic notion of ϵ-Strongly Differentially Uniform permutations (ϵ-SDU), by allowing ϵ to be a function of the security parameter k.

Definition 2. *Let $\mathcal{G} : \{0,1\}^k \times \{0,1\}^k \rightarrow \{0,1\}^k$ be a family of permutations indexed by security parameter k, and $\mathcal{A}$ be a computationally unbounded TM. Consider the following experiment $\mathsf{SDU}_{\mathcal{A},\mathcal{G}}(k)$:*

1. *$G \leftarrow \mathcal{G}$ is selected at random by choosing the key. Further, a random $R \in \{0,1\}^k$ is chosen.*
2. *$\mathcal{A}$ provides d, and receives $G(d \oplus R)$. $\mathcal{A}$ outputs a pair $\Delta x, \Delta y \in \{0,1\}^k \setminus 0$.*
3. *The output of the experiment is defined to be 1 if $G(d \oplus R) \oplus \Delta y = G(d \oplus R \oplus \Delta x)$, and 0 otherwise.*

[1] For our application, fewer rounds (e.g., 8) would provide adequate security, because the keys are refreshed frequently, and $\mathcal{A}$ would only have on the order of seconds or minutes in order to "crack" the MAC.

[2] It is easy to see that, e.g., if G is unkeyed, $\mathcal{A}$ can easily construct a forgery.

We say that $\mathcal{G}$ is $\epsilon(k)$-Strongly Differentially Uniform (ϵ-SDU for short), if for all $\mathcal{A}$, $\Pr[\mathsf{SDU}_{\mathcal{A},\mathcal{G}}(k) = 1] \leq \epsilon(k)$, where the probability is taken over the random coins used in the experiment.

It is easy to see that ϵ-SDU is a notion derivative from ϵ-DU. Indeed, Definition 1 can be cast asymptotically and in the game style, resulting in exactly the Definition 2, with the exception that $\mathcal{A}$ is not given $G(d \oplus R)$ in the corresponding experiment $\mathsf{DU}_{\mathcal{A},\mathcal{G}}(k)$. Note, the notion of ϵ-SDU is strictly stronger than that of ϵ-DU. Indeed, while unkeyed ϵ-DU functions exist [19], unkeyed ϵ-SDU functions don't. (This is because for every Δx, an ϵ-SDU $\mathcal{A}$ can output a winning Δy since he can invert the received $G(d \oplus R)$.)
We now show how to efficiently transform any ϵ-DU family into ϵ-SDU.

Lemma 1. *Let k be a security parameter, and $\mathcal{G}'$ be a keyed (or unkeyed) ϵ-DU permutation family. Let $\mathcal{G} = \{G = G' \oplus \ell_1 | G' \in \mathcal{G}', \ell_1 \in \{0,1\}^k\}$ be a family additionally keyed by uniformly chosen $\ell_1 \in_R \{0,1\}^k$. Then $\mathcal{G}$ is an ϵ-SDU permutation family, for the same ϵ.*

Démonstration. We postpone the proof to the full version of the paper. □

As a consequence of the lemma, we can construct efficient ϵ-SDU permutations from known ϵ-DU permutations, such as AES, at additional negligible cost.

3.3 ShMAC Construction

Let $r \in \{0,1\}^k$ be a nonce; in practice r may be chosen randomly for each MAC evaluation. Let $\mathcal{G}$ be a ϵ-SDU permutation family (Definition 2), where $\epsilon = \epsilon(k)$ is negligible in k. Let G be a random member of $\mathcal{G}$, selected by randomly choosing the key ℓ. Let $F : \{0,1\}^k \to \{0,1\}^k$ be chosen at random, and unknown to adversary; in practice F is to be implemented by a PRPG, such as AES.

Construction 1 *Shallow MAC is the algorithm :*

$$\mathrm{ShMAC}_\ell(r, d) = (r, G(d \oplus F(r))), \tag{3}$$

where r is a nonce.

Theorem 1. *Construction 1 is a nonce-based MAC.*

Démonstration. We postpone the proof to the full version of the paper. □

We observe that ShMAC can be executed on multiple data blocks by simple concatenation of MACs of individual blocks. This observation is motivated by the fact that efficient ϵ-SDU functions may not be readiliy available from the literature for larger data blocks. For simplicity, we state the following lemma for the case of two blocks; it can be naturally extended to any number of blocks.

Lemma 2. *Let F, G, r be as above, and let d_0 and d_1 be data blocks. Then*

$$MAC(d_0, d_1) = (r, G(d_0 \oplus F(r, 0), G(d_1 \oplus F(r, 1))) \tag{4}$$

is a nonce-based MAC.

Démonstration. We postpone the proof to the full version of the paper. □

3.4 Instantiation Considerations

We now provide some instantiation considerations. Theorem 1 is stated with respect to an ideal object — a randomly chosen function. In practice, this is implemented by means of a PRPG, and therefore Theorem 1 becomes conditional on the existence of PRPGs. Of course, this transition into the computational model improves the chances of $\mathcal{A}$ to forge the MAC, but it can be easily shown that this improvement is negligible. Note that ϵ-DU functions (and thus ϵ-SDU functions, via Lemma 1) are known to exist, and their use does not constitute an assumption.

For most applications, including our secure DRAM architecture, we envision the use AES to implement the PRPG, which is a natural choice given AES's standardization and efficiency.

As noted previously, "shortcut" 2- or 4-round versions of AES are ϵ-DU permutations. Further, *AddRoundKey*, the final phase of each AES round, implements the transformation of Lemma 1. At the same time, in many hardware implementations, the AES key schedule is precomputed, with round keys being randomly chosen. Such shortcut AES implementations satisfy the stronger ϵ-SDU requirements and are sufficient for security of MAC. In our implementation, we follow this approach.

Depending on application, desired input length of MAC may vary. In our encrypted memory system, for example, we operate on 256-bit blocks. We wish to point out several observations that apply to such usage scenarios. First, it is not necessary to use wider, e.g., 256-bit, block cipher as the PRPG F. Wider block ciphers are more expensive, since they aim to achieve strong bit mixing on the full block. For example, 256-bit Rijndael requires 14 rounds, vs the 10 rounds of its 128-bit AES sibling. In our application F is only a source of randomness, and it is sufficient to execute AES twice with corresponding adjustment of the nonce r to $(r, 0)$ and $(r, 1)$. Second, $\mathcal{G}$ must be chosen properly as well. Similarly to AES, 256-bit Rijndael achieves good bit mixing after only 4 rounds [4,3]. Alternatively, we could apply Lemma 2 and execute 128-bit $\mathcal{G}$ (e.g., AES2 or AES4) on each of the two 128-bit halves of masked data.

In our secure memory application, we choose the nonce r. Quite naturally, we choose r to be a concatenation of the address of the memory location and a global RAM transaction counter (the latter may need to be wrapped around for efficiency). This provides a simple and efficient way of generating nonces. Further, this method allows binding the memory value to the memory location, preventing replay of valid data at wrong locations[3][4]. Another advantage of this nonce choice is that the bulk of the nonce, the memory address, need not to be written to memory, as it is managed by underlying subsystems. Further, this method ensures that nonce is available before the data arrives, and thus allowing precomputation.

We discuss these and other considerations and design choices in more detail in Section 4.

4 Applications : Secure DRAM

In this section we give an overview of the design of a secure system based on System-on-a-Chip (SoC) technology, which makes use of the ShMAC construction. As noted in Section 1, all system operations (with the exception of memory transactions) take place inside the presumably secure and tamper-resistant chip. Therefore, securing the memory, which might be adversarily controlled, closes the main avenue of attack. We discuss several hardware

[3] The binding between the data and the nonce is guaranteed by the strong definition of MAC that we use. Indeed, it disallows polytime $\mathcal{A}$ to generate new nonce-tag pairs even on previously signed data.

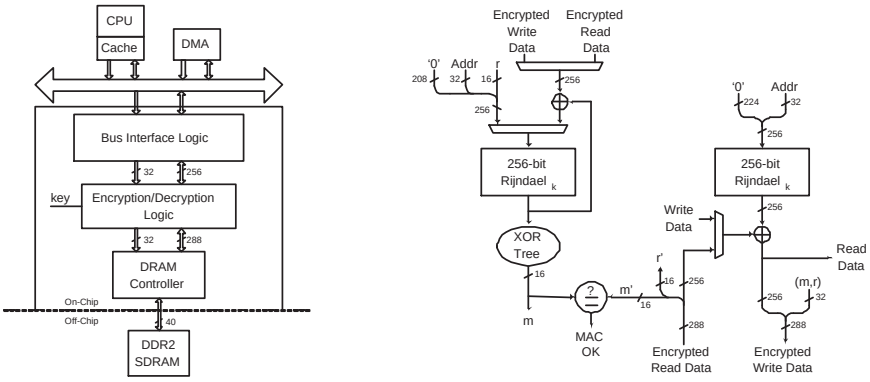
[4] Note, this does not prevent replay at the same memory location. We discuss this trade-off in Section 4.

aspects of an encrypted memory implementation using ShMAC for authentication, associated trade-offs and improvements.

In our presentation, we omit the discussion of several other aspects of the system, such as secure-boot procedures, since their design — a complex security engineering undertaking — is not directly related to the main subject of this work. We start by presenting the Encryption/Authentication Unit (EAU), its on-chip location, connectivity and relationship with other units.

4.1 Overview of Memory Encryption and Authentication

As shown in the conceptual block diagram, Figure 1(a), the EAU is interposed between a conventional DRAM controller and the interface logic that allows potential bus masters, such as CPUs and DMA engines, to access secured off-chip memory. DRAM write transactions are encrypted on the way out to DRAM and read transactions are decrypted coming back from DRAM to the SoC. During the encryption process, a MAC is generated and stored with each encrypted block of memory to ensure data integrity. During subsequent DRAM read operations, the stored MAC is compared with a newly recomputed MAC to detect corrupted off-chip memory contents.



(a) SoC system layout (b) Conceptual EAU design

FIG. 1: SoC-based Encrypted DRAM

Each MAC is associated with a fixed number of data bytes, called an *encryption block*. The encryption block is the minimal unit of data. That is, the EAU supports only block-size read or write off-chip DRAM transactions (and transparently handles creation and verification of the associated MACs). The bus interface logic handles transactions of all sizes. If a bus write transaction affects only a portion of an encryption block, the system first needs to read, decrypt and verify the unavailable bits (if any) of the encryption block from off-chip DRAM. Then the EAU merges the bits to create a full updated encryption block, before it is re-encrypted and written off-chip to DRAM.

The size of the encryption block and the number of bits in the MAC is a complex engineering trade-off. Clearly, each bit of MAC stored in DRAM is unavailable for user data and therefore

represents overhead in an encrypting memory system. The more bits of MAC are used per encryption block, the less likely it is that an adversary would be able to subvert the tamper detection process. Since MACs are stored in the same DRAM as the encryption blocks, there is also the physical and costs constraints of the DRAM data width. For most SoC systems, especially with DDR interfaces[5], the DRAM is usually 16 or 32-bits wide. Therefore, MACs with that granularity are preferred.

Similarly, the size of the encryption block is determined by the range of data sizes expected in typical SoC bus transactions. DMA transfers can generate bus transaction sizes of anything from single bytes to multi-word IP packets, but processors present a characteristic bus transaction width that corresponds to their cache line size. Choosing an encryption block size the same as the processor's cache line will efficiently support the most frequent bus transactions.

Our encrypted memory supports a physical 32-bit wide DRAM system. Encryption blocks are 256-bit wide and the associated MAC can be as short as 32 bits, while providing reasonable security. Each DRAM transaction is therefore eight 32-bit words of data followed by one or two words of MAC. This way, the memory overhead is as low as 12.5% and up to 8/9-ths of the DRAM contents is available for user storage.

Stateless vs. stateful integrity checks. In our design, the EAU is stateless. This is necessary due to severe on-chip resource restrictions. It is not hard to see that encryption and authentication process as described above exposes the system to replay attacks. For example, an adversary can replace the current contents of memory with a value that was stored in that same location previously. Similarly, an adversary can simply not update the DRAM as required by a write transaction. It is easy to see that the system will decrypt and mistakenly accept such data as valid.

Stateful operation is *necessary* to prevent such attacks. Keeping on-chip state per each memory location, however, is clearly prohibitively expensive. A natural solution is to build a Merkle tree [17] of MACs for the entire memory contents, as proposed and deployed in, e.g., [11,22,21]. However, even with the possible optimizations, maintaining such a tree of MAC values is a performance bottleneck and requires significant on-chip resources, which is unacceptable in many settings, including ours. In our system, instead of expensive Merkle-tree-based integrity checking, we use a much faster method to achieve a level of security that is sufficient for most commercial applications.

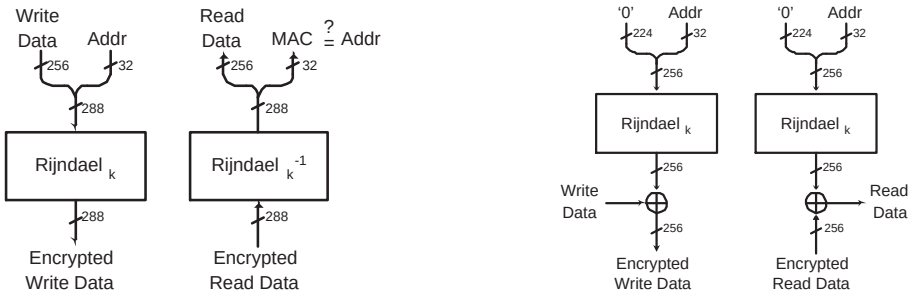
In our design, to limit the exposure to replay attacks, we propose changing memory encryption keys frequently enough so as to invalidate sufficiently stale encrypted memory state. Refreshing memory keys is easy to implement, as memory can be divided into two regions separated by a boundary address. Each region is encrypted with its own key and the refresh process is simply a matter of growing one region at the expense of the other by decrypting memory contents on one side of the boundary, re-encrypting with the other key, and then moving the boundary to reflect the new memory state. When the boundary reaches the end of memory, the old key is retired, a new key is created and the refresh process starts again. If memory keys are refreshed often enough, say, every couple of minutes, then the window of vulnerability to replay attacks is fairly narrow.

Idle-time key refreshment is *much* more efficient than maintaining a Merkle tree. We believe that frequent key expiration, and a single MAC per encryption block affords practical levels of security with much less mechanism and performance penalty, and thus is a better security/performance trade-off.

[5] DDR stands for Double Data Rate, a signaling standard that transfers data on both rising and falling clock edges. DRR2 and DDR3 are the common current DRAM interface standards.

4.2 EAU Implementation Using ShMAC

The most direct method to encrypt and authenticate off-chip memory transactions, would be to encrypt the concatenated address and data[6]. This would produce a 288-bit encrypted memory write value, as shown in Figure 2(a). However, this method serializes the encryption process with memory write operations and, more importantly, adds decryption delay to the already performance-limiting DRAM read latency. Additionally, this scheme requires both encrypt logic and decrypt logic, which is unacceptable for FPGA implementations.



(a) Direct method (b) Low-latency method

FIG. 2: Encryption/authentication methods

Motivated by low-latency and small footprint requirements, we prefer a different encryption approach, shown on Figure 2(b), and separate MAC generation from the data encryption process. Figure 1(b) illustrates a conceptual EAU design, described below[7]. A 256-bit pad is generated by Rijndael encryption of the address[8]. The pad is then XORed with the Write data to produce the encrypted result. Since XOR is its own inverse, the same encryption function can be used for both encryption as well as decryption. While encryption remains serialized with the DRAM write operation, the pad calculation can start as soon as the address is available early in the bus transaction. More importantly, for performance-critical read operations, the pad calculation can occur in parallel with DRAM read latency. Once encrypted DRAM data is available, a single XOR operation is the only additional delay incurred by decryption. As a result, decrypted data is returned to the processor with negligible delay.

MAC generation proceeds as follows. The PRP F of Construction 1 is achieved by running full Rijndael on the address concatenated with a nonce r. The nonce value can be a global counter that increments with each memory write transaction. We stress that there is no need for expensive pseudorandom generation of the nonce. Note that this first step of the MAC calculation can start as soon as an address is available, simultaneously with the encryption

[6] Additional redundant data can be added under the encryption, if stronger integrity checks are desired.

[7] Reasonable security parameter sizes were included in Figure 1(b) for concreteness, however, their values should be evaluated for specific instantiations. See discussion below.

[8] We note that the 256-bit pad can be more efficiently generated by two parallel 128-bit AES encryptions in fewer rounds. We omit this, as well as other natural optimizations, for the sake of clarity.

process. The Rijndael output is then XORed with the encrypted data and the same Rijndael data path is reused to compute G of ϵ-SDU family $\mathcal{G}$. In our implementation, G is a four-round Rijndael evaluation[9]. The output of G is collapsed via an XOR tree to a value m, which is concatenated with the original unmodified nonce r to form the MAC written to DRAM — this is the ShMAC output.

In contrast with the decryption process, the MAC verification for memory read operations must first wait for the DRAM latency in order to acquire the original nonce r, which is stored off-chip. Once data and MAC arrive, F is computed on the address appended with r (14 rounds of 256-bit Rijndael). This value is then XORed with the encrypted read data and the same Rijndael data path is reused to compute G, which consists of four rounds of Rijndael. The XOR tree collapses the result to generate m, which is compared with m', the value of the just-read, off-chip MAC. If they match, the memory read operation is considered uncorrupted.

Note, MAC verification can only start after the original MAC value is read and much later than the actual decryption process, which means that data would have already been returned to the processor before the MAC is verified. We can afford this delay because in our application we consider MAC failure to be so dire that the system effectively resets and discards any use of the corrupted data. Thus, we do not need to implement any recovery mechanisms, such as rollbacks.

Trade-offs and design choices. Due to the unacceptable cost of tree-based integrity checking, it was our decision to use weaker but much more efficient authentication, which allows replay attacks within a small window (e.g., one to several minutes). We believe this is a reasonable compromise. Next, we argue that our authentication approach effectively limits the forger to replay attacks.

Performance considerations require use of short MACs. We first argue that even 16-bit security is sufficient in many practical security applications[10]. (Of course, this parameter would need to be evaluated for each concrete system instantiation, using the following discussion as a guideline.) Indeed, on average, it would take the adversary 2^{15} attempts to forge just one memory block. Note that in our system each unsuccessful attempt would be followed by a forced reboot (a natural reaction to a break-in attempt), which might take around a minute to complete. This means that forging a single block would take an expected 20 days of continuous attacks; forging even two blocks (expected 2^{30} attempts) is infeasible. Thus, attackers are likely to use other attack avenues, such as exploiting the replay permissiveness.

Achieving 16-bit security requires the use of MACs of greater length, since the ShMAC output includes a nonce. In our system, the ShMAC nonce consists of the concatenation of the address and r. We first observe that nonces for different memory locations would never collide; however, nonces may collide within the same memory locations. If many collisions occur, the adversary may eventually accumulate some useful information about G. We mitigate this threat with periodically refreshing F and G (by changing their keys). As an additional disadvantage to the adversary, he does not learn the full value of G's, but only a fraction of it. Thus, we believe that a choice of length for r in the 16–48 bits range would be appropriate for most applications.

Finally, we also note the following trade-off regarding the encryption approach. Because we encrypt by XORing data with the pad derived from the memory address, the adversary would be able to track changes in data stored in memory. Specifically, when two data blocks d_1, d_2

[9] As discussed in Section 3.4, we alternatively could use parallel execution of two instances of 2- or 4-round AES.

[10] Of course, by 16-bit security we mean that the probability of a polynomial-time adversary forging a MAC is $(\frac{1}{2})^{16}$, and not that it takes 2^{16} operations to break it.

are encrypted with the same key and stored in the same memory location, the adversary is able to compute $d_1 \oplus d_2$ as the XOR of their encryptions. Again, we believe that in many commercial applications, this weakness, mitigated in particular by frequent key refreshes, does not significantly help reverse engineering and other hostile analysis and interference. Recall that the use of this method for encryption is critical in minimizing read and write latencies. This concludes the description of our encrypted DRAM architecture using ShMAC.

5 Related Work on Fixed Input Length MAC

The main application we consider – hardware memory authentication – greatly benefits from datapath latency reduction. At the same time, in severely constrained environments we are considering (e.g., FPGA), we cannot afford the luxury of using generic MAC schemes, and must be specific with the choices of building blocks. We impose strict limits on both data-dependent computation and precomputation time and chip surface area.

Some details and applications of the specific property of MAC precomputation have been discussed in the literature (e.g., [15]), although, to our knowledge, not in the severely restricted environments that we consider. In this section, we overview previous work on message authentication, with the emphasis on precomputation. We discuss and clarify the relationship between the building blocks, clarify the terminology and explicate efficient constructions.

Recall, we are interested in authenticating 256-bit data blocks. The most direct approach to MAC is to simply encrypt (e.g., with a blockcipher, such as AES) the data concatenated with the address, and possibly some redundancy. However, this solution is unsatisfactory since it does not lend itself to precomputation, and, further, requires both encryption and decryption hardware.

Before discussing previous work in more detail, we recall some definitions. Let $H : K \times X \rightarrow Y$ be a function family, indexed by the key $k \in K$. A Universal Hash Function (UHF), or universal$_2$, H guarantees that $\forall x_1 \neq x_2 \in X, \Pr_k[H_k(x_1) = H_k(x_2)] \leq \frac{1}{|Y|}$. That is, no pair of preimages is mapped into the same value by more than one $|Y|$-th of the functions. A stronger notion of Strongly Universal (SU) H requires that $\forall x_1 \neq x_2 \in X, \forall y_1, y_2 \in Y, \Pr_k[H_k(x_1) = y_1 \wedge H_k(x_2) = y_2] = \frac{1}{|Y|^2}$. In other words, H_k maps all distinct x_1, x_2 independently and uniformly.

One of the most celebrated MAC schemes, and also one that naturally allows precomputation, was proposed by Wegman and Carter [24]. Extending the authors' previous work on UHF families, [24] introduced the notion of SU hash families, and showed that $MAC_{k,r}(m) = H_k(m) \oplus r$ is an unconditionally secure MAC, where H is SU, r is one-time pad, and k is a random index into the family H.

Stinson [20] formalized the notion of ϵ-Almost SU (ASU), a more general class of functions usable with the Wegman-Carter MAC construction. As the name suggests, ASU functions simply allow less strict bounds on the probability guaranteed by SU. Stinson also showed how to combine a (faster) UHF with an ASU function to obtain a faster ASU function. Brassard [1] pointed out that a pseudorandom generator could be used in place of one-time pad. Krawczyk [14] noticed that ϵ-Almost XOR Universal (AXU), a weaker than ASU function family is sufficient for Wegman-Carter MAC. (Recall, H is ϵ-AXU, if $\forall x_1 \neq x_2 \in X, \forall c \in Y, \Pr_k[H_k(x_1) \oplus H_k(x_2) = c] \leq \epsilon$. Krawczyk [14] called this notion *otp-secure*, but AXU is the more frequently used term today.)

Following these fundamental results, a lot of work went into the design of efficient universal, AU, ASU and AXU functions. Most of the research concentrated on software-efficient functions, i.e., those that could take advantage of CPU's instruction sets which, in particular,

includes multiplication. Unfortunately, algebraic solutions are not acceptable in our setting, due to latency and cost of hardware implementation of multiplication.

In fact, only acceptable solutions would reuse the circuitry of the PRFG to generate the pad r and to evaluate H. Our solution, presented in Section 3 does just that. Alternatively, a MAC scheme with very similar performance to our scheme can be extracted from a large volume of previous literature on the subject. Several papers contribute pieces of the total solution, but, to our knowledge, none explicitly states it ; further, several sources use conflicting terminology.

Firstly, we point out that neither UHF nor AU functions are sufficient for Wegman-Carter MAC security. This is because they do not guarantee that an offset in the argument will not result in an unpredictable offset in the value of H_k. For example, an identity function is a UHF, but clearly a Wegman-Carter MAC based on it is easily forged. We note, however, that UHF and AU are often used in MACs for efficiency reasons, but only as *part* of the function H ; a stronger ASU component is additionally required in H [20]. Further, some sources, e.g., [25], blend the notions of UHF and SU, defining UHF as SU.

Therefore, although it was previously observed, e.g., in [18], that it is possible to obtain AU functions from four-round AES, such results are not applicable for our uses of Wegman-Carter MAC. To our knowledge, the only explicit AXU construction from an ϵ-DU function recently appeared in [12]. In particular, [12] uses AXU derived from a 4-round AES in the Wegman-Carter MAC. However, their MAC construction ([12], Algorithm 1) generates fresh keys for H and the pad r for each MAC execution, an unacceptable overhead for our setting. We note however, that in our setting, the keys of H can be reused, which would bring the resource requirements down to that of our construction of Section 3.

Références

1. Gilles Brassard. On computationally secure authentication tags requiring short secret shared keys. In *Advances in Cryptology – CRYPTO 82*, pages 79–86. Springer-Verlag, 1983.
2. Benoit Chevallier-Mames, David Naccache, Pascal Paillier, and David Pointcheval. How to disembed a program? Cryptology ePrint Archive, Report 2004/138, 2004. `http://eprint.iacr.org/`.
3. Joan Daemen. Annex to AES proposal Rijndael. chapter 5. propagation and correlation. Available at `http://www.iaik.tugraz.at/Research/krypto/AES/old/~rijmen/rijndael/PropCorr.PDF`.
4. Joan Daemen and Vincent Rijmen. AES proposal : Rijndael. Available at `http://www.iaik.tugraz.at/Research/krypto/AES/old/~rijmen/rijndael/rijndaeldocV2.zip`.
5. Joan Daemen and Vincent Rijmen. Understanding two-round differentials in AES. In *SCN*, pages 78–94, 2006.
6. Guillaume Duc. Cryptopage. Master's thesis, ENST, Bretagne, June 2004.
7. Reouven Elbaz, David Champagne, Ruby B. Lee, Lionel Torres, Gilles Sassatelli, and Pierre Guillemin. Tec-tree : A low-cost, parallelizable tree for efficient defense against memory replay attacks. In P. Paillier and I. Verbauwhede, editors, *CHES 2007*, pages 289–302, Berlin Heidelberg, 2007. LNCS 4727, Springer-Verlag.
8. Reouven Elbaz, Lionel Torres, Gilles Sassatelli, Pierre Guillemin, Michel Bardouillet, and Albert Martinez. A parallelized way to provide data encryption and integrity checking on a processor-memory bus. In *DAC 2006*, pages 506–509, San Francisco, California, USA, July 24-28 2006. ACM.

9. Blaise Gassend, G. Edward Suh, Dwaine Clarke, Marten Van Dijk, and Srinivas Devadas. Caches and hash trees for efficient memory integrity verification. In *In 9th Intl. Symp. on High Performance Computer Architecture*, 2003.
10. A. Gonsalves. Hackers report breaking Phone 2.0. InformationWeek, March 12, 2008.
11. Guerney D. H. Hunt. Secure processors for secure devices and secure end-to-end infrastructure. Available at `http://www.research.ibm.com/jam/secure-processors5-30-06.pdf`.
12. Goce Jakimoski and K. P. Subbalakshmi. On efficient message authentication via block cipher design techniques. In *Advances in Cryptology – ASIACRYPT 2007*, volume 4833, pages 232–248, 2007.
13. L. Keliher and J. Sui. Exact maximum expected differential and linear cryptanalysis for two-round Advanced Encryption Standard. IET Information Security , Vol. 1, No. 2, pp. 53-57, June 2007.
14. Hugo Krawczyk. LFSR-based hashing and authentication. In *Advances in Cryptology – CRYPTO 94*, pages 129–139, London, UK, 1994. Springer-Verlag.
15. Hugo Krawczyk, Mihir Bellare, and Ran Canetti. RFC2104 - HMAC : Keyed-hashing for message authentication. `http://www.faqs.org/rfcs/rfc2104.html`. Retrieved Sept. 17, 2008.
16. David Lie, Chandramohan A. Thekkath, Mark Mitchell, Patrick Lincoln, Dan Boneh, John C. Mitchell, and Mark Horowitz. Architectural support for copy and tamper resistant software. In *Architectural Support for Programming Languages and Operating Systems*, pages 168–177. ACM, 2000.
17. Ralph Merkle. *Secrecy, authentication, and public key systems.* PhD thesis, Stanford Univeristy, 1979.
18. Kazuhiko Minematsu and Yukiyasu Tsunoo. Provably secure MACs from differentially-uniform permutations and AES-based implementations. In *FSE*, pages 226–241, 2006.
19. Kaisa Nyberg. Differentially uniform mappings for cryptography. In *Advances in Cryptology – EUROCRYPT 93*, volume 765, pages 55–64. Springer-Verlag New York, Inc., 1994.
20. Douglas R. Stinson. Universal hashing and authentication codes. In *Advances in Cryptology – CRYPTO 91*, pages 74–85, London, UK, 1992. Springer-Verlag.
21. G.E. Suh, C.W. O'Donnell, and S. Devadas. Aegis : A single-chip secure processor. *Design and Test of Computers, IEEE*, 24(6) :570–580, Nov.-Dec. 2007.
22. Gookwon Edward Suh. *AEGIS : A Single-Chip Secure Processor.* PhD thesis, MIT, 2005.
23. Trusted Computing Group. *TCG Specification Architecture Overview*, revision 1 edition, July 2007.
24. M. N. Wegman and J. L. Carter. New hash functions and their use in authentication and set equality. In *J. Comput. System Sci. 22*, pages 265–279, 1981.
25. Eric W. Weisstein. "Universal hash function." From MathWorld–a Wolfram web resource. `http://mathworld.wolfram.com/UniversalHashFunction.html`. Retrieved Sept. 17, 2008.

Environnement matériel de confiance et sécurité des protocoles distribués

Carlos Aguilar Melchor, Céline Burgod, et Julien Iguchi-Cartigny

XLIM-UMR CNRS 6172,
83 rue d'Isle, 87000 LIMOGES, FRANCE
carlos.aguilar,celine.burgod,julien.cartigny@xlim.fr

Résumé Un défi dans la conception de protocoles distribués sécurisés où des processus indépendants sont intégrés dans un environnement matériel de confiance vient du fait que les canaux physiques de communication sont entièrement sous le contrôle de leur hôte. Un hôte hostile est en mesure d'intercepter chacun des messages en entrée et en sortie de son environnement matériel de confiance, conduisant potentiellement à des erreurs dans les calculs réalisés à l'intérieur de l'environnement matériel de confiance. Dans cet article, nous étudions ce problème puis nous nous intéressons à la détection par le matériel de confiance d'actes non-conformes exhibés par un hôte dans les couches de transport les plus basses.

Mots-clés : Matériel de confiance, protocole distribué, sécurité

1 Introduction

De nos jours, les équipements matériels de confiance résistants à la manipulation (Trusted Hardware ou TH) sont utilisés dans de nombreuses applications pour lesquelles des garanties fortes dans l'exécution d'un processus ou la préservation d'informations confidentielles sont requises. Leur principale propriété est qu'ils fournissent un environnement autonome d'exécution et de stockage de données ne pouvant pas être inspecté ou modifié. En d'autres termes, un attaquant n'est plus capable d'intervenir directement sur les données stockées, sur les calculs réalisés, ou sur le contenu des messages générés par un TH. Ainsi, des milliards de cartes à puce ont été déployées pour par exemple sécuriser le système Global System for Mobile Communications (GSM), les systèmes de transactions bancaires, les services d'authentification et de contrôle d'accès à des utilisateurs.

Le matériel de confiance est souvent considéré comme un outil de sécurisation infaillible contre les comportements byzantins. Cependant, il est souvent mal compris à quel point cela est faux dans les protocoles distribués où le canal physique de communication ne peut généralement pas être sécurisé. En nous plaçant dans le cas habituel où un TH requiert explicitement les ressources de son hôte pour communiquer avec d'autres TH, un attaquant peut supprimer, fabriquer, modifier ou rejouer des messages. Dans cet article, nous présentons ce problème et nous nous intéressons à la détection par le matériel de confiance d'actes non-conformes aux protocoles de communication dans les couches de transport les plus basses. L'étude met en relief que la détection est possible, mais limitée, et donc qu'il est illusoire de considérer que l'utilisation d'un matériel de confiance suffit à sécuriser un protocole distribué, même s'il peut en améliorer l'auditabilité.

Le reste de cet article est organisé comme suit. La section suivante présente les problèmes de sécurité intrinsèques au matériel de confiance puis nos hypothèses sur le réseau. La section 3

décrit un protocole d'identification par le matériel de confiance du comportement d'un hôte dans les opérations de bas niveau d'émission et de réception de messages, puis discute ses limitations. La section 4 présente une application dans le cadre des réseaux ad hoc. La section 5 évalue les coûts introduits par notre système en termes de volume de données, de nombre de messages et d'espace mémoire requis. Dans la section 6, les travaux en lien avec notre étude sont présentés. Finalement, la conclusion rend compte des résultats obtenus.

2 Modélisation du système

Nous étudions le problème de l'exécution de protocoles distribués faisant interagir de multiples nœuds au moyen d'échanges de messages. Dans notre modèle, chaque processus autonome s'exécute dans un équipement matériel de confiance (TH). En pratique, ce TH peut être une carte à puce ou un co-processeur sécurisé.

2.1 Relations entre TH et hôtes de non-confiance

Nous considérons que les échanges de messages entre TH sont sécurisés au moyen d'outils cryptographiques. Une signature numérique est calculée (à partir d'une clé cryptographique maintenue secrète) puis ajoutée à tout message généré par un processus avant de sortir du TH. Cette signature est ensuite vérifiée par le TH de tout nœud récepteur d'un tel message afin d'en authentifier l'origine : tout message présentant une signature correspondante valide est traité, sinon il est rejeté.

En revanche, les canaux physiques de communication ne sont pas sûrs. En effet, un TH n'est pas considéré comme étant un nœud réseau communicant autonome. Il requiert les ressources matérielles de communication de son hôte pour pouvoir interagir avec d'autres TH. Un hôte a donc pour responsabilité de relayer les messages, à savoir d'émettre les messages générés par son TH sur le réseau, puis de retransmettre les messages reçus sur son interface de communication à son TH. La Figure 1 illustre le flux de messages entre TH et hôtes à travers le réseau.

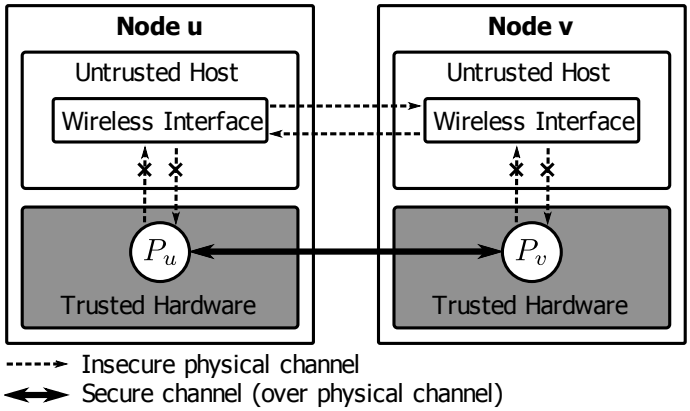


FIG. 1: Vulnérabilités des canaux de communication

2.2 Profils de l'hôte

Le TH apporte des garanties explicites en terme d'intégrité sur le contenu des messages. Puisque seuls les messages signés sont considérés comme valides, un hôte n'est plus en mesure de fabriquer ou modifier directement les informations encapsulées dans les messages échangés entre TH. La principale vulnérabilité du système vient du fait que les nœuds hôtes peuvent intercepter les messages en entrée et en sortie de leur TH. Ces nœuds hôtes peuvent être classifiés selon la façon dont ils agissent dans leur rôle d'intermédiaire. Nous définissons trois types de comportement :

- Un hôte a un comportement conforme lorsqu'il réalise sans distinction toutes ses opérations d'émission (soit d'émettre tous les messages générés par son TH sur le réseau), et toutes ses opérations de réception (soit de transmettre tous les messages en provenance du réseau à son TH) ;
- Un hôte a un comportement non-conforme dans l'opération d'émission lorsqu'il omet d'émettre sur le réseau un message généré par son TH ;
- Un hôte a un comportement non-conforme dans l'opération de réception lorsqu'un hôte omet de transmettre à son TH un message reçu sur son interface de communication.

2.3 Modèle du réseau

Nous supposons que les nœuds sont statiques, qu'ils communiquent au moyen de liens sans fil et que les transmissions sont fiables, à savoir qu'il n'y a pas de perte de messages due à des collisions, des interférences, etc. Nous partons de l'hypothèse que les liens physiques de communication entre les nœuds sont symétriques. En d'autres termes, si un nœud u est capable de recevoir les messages émis par un nœud v, alors v est également capable de recevoir les messages émis par u. Nous dénotons $\vartheta(u)$, l'ensemble des nœuds à portée physique de communication du nœud u. L'émission d'un message m d'un hôte u vers un hôte v est dénotée $u \rightarrow v : m$, et inversement la réception d'un message est dénotée $u \leftarrow v : m$.

Par ailleurs, chaque TH maintient secret un identifiant de nœud qui est utilisé dans les phases d'échanges de messages.

2.4 Sécurité des protocoles distribués

Cette première analyse met en évidence le fait qu'il n'est pas possible d'assurer le fonctionnement correct d'un protocole distribué uniquement en sécurisant son exécution sur chacun des nœuds participants, car la possible suppression des messages sera critique dans un grand nombre de cas. Par contre, nous pouvons essayer de voir à quel point il peut y avoir une détection des comportements non-conformes qui puisse inciter les hôtes à collaborer, réduisant ainsi considérablement les attaques contre ces protocoles. C'est ce que nous présentons dans la prochaine section.

3 Détection des comportements non-conformes

Notre objectif est d'évaluer, au niveau des couches de transport les plus basses, la capacité d'un nœud hôte à jouer son rôle d'intermédiaire, à savoir d'émettre les messages générés par son TH, puis de recevoir les messages en provenance des autres nœuds présents dans son voisinage.

Dans cette section, nous étudions dans quelle mesure un TH peut permettre de déterminer le comportement d'un hôte au regard de ces deux opérations pour l'utiliser à un plus haut niveau dans un système d'évaluation de la qualité des liens de communication. Pour ce, nous proposons un protocole de contrôle fondé sur des accusés de réception (ACK). Un tel système d'évaluation pourra être utilisé tel quel ou à son tour faciliter la mise en place d'un système de réputation ou récompense.

3.1 Description du protocole

Dans notre approche, nous exploitons l'omnidirectionnalité des transmissions pour identifier le comportement d'un hôte dans les opérations d'émission et de réception d'un message. Dans notre modèle, si un hôte u émet un message m sur le réseau, alors tous les hôtes présents dans son voisinage physique de communication doivent le recevoir ($\forall v \in \vartheta(u), v \leftarrow u : m$). Nous considérons par ailleurs qu'au moins un nœud dans le voisinage d'un émetteur adopte un comportement conforme. Ce nœud servira de témoin en cas d'ambiguïté dans l'évaluation du comportement d'un autre nœud.

Dans le cas où un hôte u est entouré uniquement d'hôtes non-conformes, le TH de u n'aura aucun témoin pouvant lui donner des indices sur un éventuel comportement non-conforme de son hôte. Il est possible que des protocoles élaborés permettent de détecter un comportement non-conforme dans ces situations, mais nous n'explorons pas ces possibilités. En effet, l'esprit de nos travaux est de proposer une méthode locale et simple au niveau physique qui donne des bonnes propriétés émergentes dans le système global. Il est peu probable qu'il soit possible, avec une évaluation locale, de détecter des comportements non-conformes quand un hôte est entouré d'hôtes non-conformes, mais d'un autre côté, une telle approche a tous les avantages de robustesse, mise à l'échelle, etc. des systèmes émergents. L'impact des groupes non-conformes devra être étudié dans chaque application et, éventuellement, un système de détection à plus haut niveau de tels groupes devra être mis en place.

Le fonctionnement du protocole est comme suit. Lorsque le TH_u génère un message (contenant des informations relatives à l'identifiant du nœud d'origine ainsi qu'un identifiant unique de message) et le transmet à son hôte u, il passe dans un état d'attente d'un ACK (contenant l'identifiant de l'origine et du message initialement émis, ainsi que l'identifiant du nœud accusant la réception). Remarquons que seul un TH est en mesure de générer des ACK valides et que toute génération est liée à une réception au préalable d'un message.

L'hôte u a alors deux choix possibles, soit transmettre le message sur le réseau soit le supprimer. Selon l'hypothèse qu'au moins un nœud est conforme dans le voisinage de u, si aucun ACK correspondant à l'émission en attente de validation n'est reçu par le TH_u, le comportement de l'hôte u est détecté comme étant non-conforme dans l'opération d'émission. L'opération d'émission réalisée par l'hôte u est validée à partir du moment où le TH_u reçoit au moins un ACK en provenance d'un nœud quelconque. La raison de ce choix est que des hôtes non-conformes pourraient omettre de transmettre des ACK afin de faire faussement accuser des nœuds comme étant de mauvais émetteurs par leur TH.

Une fois l'opération d'émission validée, le succès ou l'échec des nœuds voisins de l'hôte u dans l'opération de réception peut être évalué par le TH_u. Ce dernier passe dans un état d'attente d'ACK en provenance de l'ensemble des TH des nœuds présents dans le voisinage de son hôte. Ainsi, tout hôte $v \in \vartheta(u)$ pour lequel le TH_u ne reçoit pas d'ACK est détecté comme non-conforme dans l'opération de réception.

La Figure 2 représente les différents résultats d'évaluation par un TH_u selon le type de comportement adopté soit par l'hôte u initialement émetteur d'un message, soit par un hôte récepteur présent dans son voisinage. Dans cette représentation, nous considérons que le

voisinage de u est composé de deux types d'hôte : un hôte récepteur conforme (dénoté c) et un autre non-conforme (dénoté m). La racine de l'arbre caractérise la réception par l'hôte u d'un message de son TH_u devant être émis sur le réseau, et chacune des branches descendantes est numérotée soit par un entier $n > 0$ pour représenter le fait qu'une ou plusieurs opérations (d'émission ou de réception) ont bien été réalisées, soit par $n = 0$ dans le cas contraire.

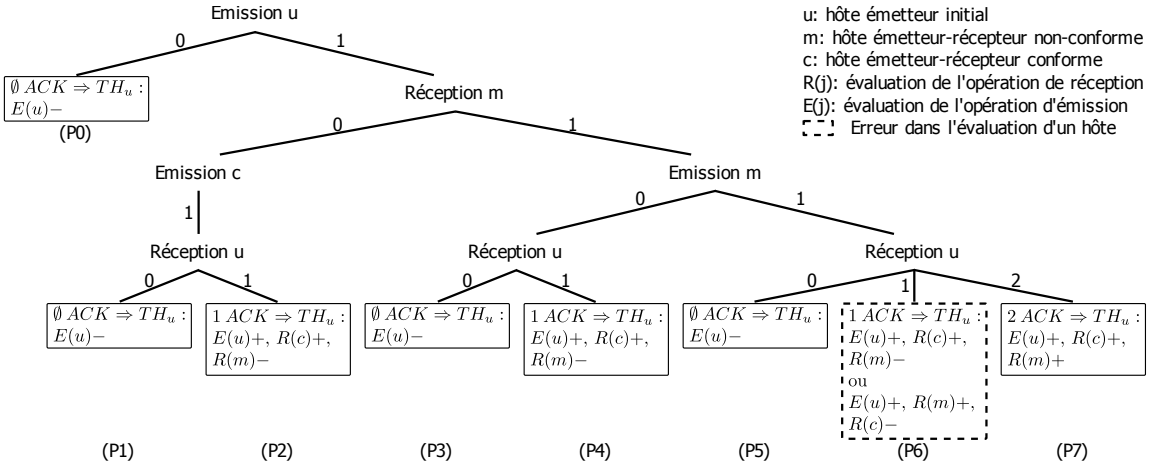


FIG. 2: Evaluation du comportement des nœuds hôtes par le TH du nœud u

Nous pouvons constater que dans la plupart des cas, le comportement des hôtes est correctement identifié par le TH_u. Cependant, nous observons qu'une erreur de diagnostic par le TH_u dans l'évaluation du comportement exhibé par un hôte voisin au regard de l'opération de réception est possible à partir du moment où l'hôte u est hostile et qu'il supprime un des multiples ACK qu'il reçoit (celui de l'hôte c ou m, chemin $P6$ de la Figure 2).

Comme nous le verrons dans la section 3.2, cette erreur d'évaluation peut se reproduire par d'autres moyens par les hôtes non-conformes et met en relief certaines limitations dans l'évaluation des comportements. Dans la section 3.3, nous présenterons un compromis permettant de réaliser des évaluations moins précises mais néanmoins utiles, en particulier sur la qualité des liens.

3.2 Détournement du protocole de contrôle

Le mécanisme d'accusé de réception permet à un TH de conclure positivement ou négativement sur la réalisation d'une opération d'émission par son hôte, puis d'autre part de conclure positivement ou négativement sur la réalisation d'une opération de réception pour tous ses hôtes voisins. Cependant, nous sommes dans l'incapacité de définir sous quelle forme cette opération d'émission est conduite par un hôte. Nous ne pouvons actuellement pas faire la distinction entre une émission en diffusion locale (soit $u \rightarrow \vartheta(u) : m$) et une émission ciblée (soit $u \rightarrow v : m$). Par conséquent, des attaques visant à tromper le mécanisme d'évaluation du comportement d'un hôte sont possibles soit lorsque des hôtes hostiles forment une coalition, soit lorsqu'un hôte embarque plusieurs TH, soit lorsqu'un hôte utilise une antenne directionnelle pour émettre ses messages.

Dans cette étude de cas, nous considérons un hôte hostile u émettant ses messages sur un canal de communication confidentiel. Ce canal peut par exemple être obtenu par chiffrement des messages au moyen d'un secret connus uniquement des membres de sa coalition. Il en découle que seuls les hôtes appartenant à la coalition de u seront en mesure d'interpréter, de faire traiter par leur TH, puis d'accuser la réception de ce message. À travers cette attaque, l'hôte u parvient à tromper son TH dans la fonction d'évaluation du comportement de tous les hôtes conformes présents dans son voisinage au regard de l'opération de réception, et ce, sans être détecté comme étant non-conforme.

En considérant qu'un nœud hôte embarque deux TH, le même type d'attaque peut être mené.

Contrairement à une antenne omnidirectionnelle pour laquelle les émissions et les réceptions s'effectuent dans toutes les directions (360°), une antenne directionnelle est une antenne grâce à laquelle les angles d'émission et de réception peuvent être paramétrés. Grâce à ce type de matériel de communication, un hôte hostile peut cibler les hôtes récepteurs de ses messages et les mêmes effets peuvent être atteints.

3.3 Evaluation de la qualité d'un lien

Alors qu'il semble difficile d'identifier de manière précise le comportement des hôtes dans leurs opérations d'émission et de réception, les informations collectées par les TH peuvent être exploitées pour déterminer la qualité d'un lien. En effet, dans tous les cas observés, soit un TH reçoit un accusé de réception d'un nœud voisin auquel cas leur lien peut être considéré comme opérationnel, et dans le cas contraire, leur lien est défaillant. Bien que limitées, ces informations peuvent par exemple servir à des protocoles de routage qui se fondent sur une évaluation de la qualité des liens pour construire des routes fiables entre des nœuds tels que Associativity-Based long-lived Routing protocol (ABR) [5]. Bien sûr, il est aussi possible de se servir d'une évaluation fiable de la qualité des liens pour mettre en place des systèmes de réputation ou de récompense basés sur ces informations.

4 Application dans les réseaux ad hoc

Dans cette section, nous montrons comment l'environnement matériel de confiance peut être utilisé pour évaluer la confiance d'un lien entre deux hôtes voisins et ainsi conduire à la sécurisation d'une infrastructure de communication au niveau transport. Nous prenons l'exemple d'un réseau formé uniquement de nœuds statiques, capables de communiquer par le biais d'ondes radio, et dans lequel les communications ne transitent pas par des équipements centraux dédiés (points d'accès, routeurs) mais où chaque nœud sert de relais des messages. Selon ce mode de fonctionnement, un nœud peut communiquer soit directement avec les autres nœuds s'ils sont dans sa portée de transmission, soit avec des nœuds distants (hors de portée) en utilisant des nœuds intermédiaires en tant que relais des messages (voir Figure 3). Ainsi, lorsqu'un nœud (dénoté A) émet en tant que source un message m, tous les nœuds présents dans son voisinage doivent le recevoir (nœud B), puis le retransmettre afin qu'il atteigne tous les autres nœuds du réseau (nœud E). De tels réseaux sont plus généralement connus sous le nom de réseaux ad hoc.

Ces réseaux sont particulièrement adaptés à des environnements où l'installation d'une infrastructure n'est pas appropriée, soit pour des raisons économiques soit pour des raisons physiques (zone géographique à accès difficile ou dévastée). Par ailleurs, puisqu'ils peuvent être déployés rapidement et avec très peu d'interventions humaines, ils peuvent par exemple être utilisés pour former un réseau de communication de secours lorsqu'une infrastructure est détruite suite à une catastrophe naturelle.

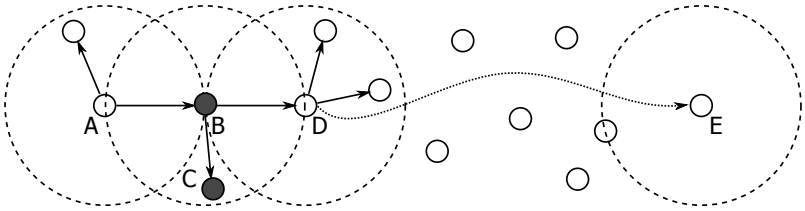


FIG. 3: Modèle de communication entre les nœuds dans un réseau ad hoc

Nous avons vu dans la section 2 qu'il ne suffisait pas d'exécuter un protocole au sein d'un environnement matériel de confiance pour en assurer le bon fonctionnement. Par exemple, dans le contexte des réseaux ad hoc, un nœud peut sans effort perturber le bon fonctionnement du réseau en ne retransmettant pas les messages destinés à d'autres nœuds du réseau sur son interface de communication. Les conséquences introduites par ce type de comportement sont généralement des pertes de messages, des pertes de connectivité, et des augmentations dans les délais d'acheminement des messages de bout en bout.

Dans la suite de cette section, nous montrons dans quelle mesure il est possible de détecter le mauvais comportement d'un hôte grâce à un mécanisme d'acquittement des messages. À partir de la Figure 3, nous étudions plus particulièrement trois scénarii possibles d'attaques dans les interactions entre hôtes et TH. Un premier dans lequel l'hôte B omet de transmettre à son TH_B le message m en provenance de l'hôte A, un second dans lequel l'hôte B omet de retransmettre sur le réseau le message m à ses voisins, et un dernier cas dans lequel l'hôte B et C forment une coalition. Notons que les paragraphes suivants n'ont pas pour vocation de traiter dans leur globalité tous les cas possibles d'attaques, mais seulement d'en détailler quelques uns afin de faciliter la compréhension des mécanismes de sécurité proposés.

4.1 Retransmission d'un message de l'hôte vers son TH

Dans ce premier exemple, l'hôte A émet en diffusion locale un message m généré par son TH_A. Nous considérons que l'hôte B omet de retransmettre à son TH_B le message m. D'un point de vue local, le résultat de cette action est que TH_A ne recevra aucun acquittement pour m signé par le TH_B en retour. Il s'agit du chemin $P2$ de l'arbre décrit Figure 2. Le degré de confiance accordé par le TH_A dans le lien entre le nœud A et B sera alors diminué.

4.2 Retransmission d'un message du TH vers le réseau

Dans ce scénario, nous considérons que l'hôte B reçoit le message m en provenance de A et le transmet vers son TH_B. Cette action entraîne la génération par le TH_B d'un message ACK ainsi que le passage dans un état d'attente d'un acquittement pour le message m en provenance de l'ensemble des nœuds présents dans son voisinage local (à l'exception du nœud A, initialement émetteur du message). En considérant que l'hôte B ne retransmet ni ACK (destiné au nœud A) pour le message m, ni le message m (destiné à l'ensemble de ses voisins), alors le degré de confiance dans le lien entre A et B du point de vue du TH_A sera diminué (chemin $P4$ de l'arbre Figure 2) et le degré de confiance dans le lien entre B et l'ensemble de ses voisins du point de vue du TH_B sera diminué (chemin $P0$ de l'arbre Figure 2).

4.3 Coalition de nœuds

Nous prenons un dernier exemple dans lequel les nœuds B et C forment une coalition. Dans ce scénario, l'hôte B reçoit le message m en provenance de A sur son interface de communication et le transmet vers son TH_B. Contrairement à l'exemple précédent, le message m est retransmis par l'hôte B. Cependant cette retransmission est réalisée dans des conditions particulières, à savoir uniquement vers le nœud C par le biais d'un canal de communication confidentiel. Le résultat direct de cette action est que du point de vue du TH_B, puisque aucun nœud présent dans le voisinage de B (à l'exception de C) n'est en mesure d'émettre un acquittement pour m (car non interprétable par ces derniers), le degré de confiance dans leur lien avec B sera diminué.

Des valeurs de confiance dans les liens sont ainsi construites par le TH d'un nœud, et ce, avec chacun des nœuds présents dans son voisinage. Ces valeurs évoluent à partir des messages d'acquittement reçus pour chaque émission réalisée. Dans les futurs échanges, ces valeurs de confiance pourront être utilisées afin d'aider la prise de décision locale sur les chemins à emprunter, apportant ainsi plus de fiabilité dans l'opération d'acheminement des messages de bout en bout. Par exemple, tout lien présentant un faible niveau de confiance pourra être évité dans l'établissement d'un chemin.

5 Analyse théorique de la complexité

Puisque plusieurs facteurs tels que la fonction de hachage cryptographique, la taille de la signature numérique et les capacités de calcul de l'environnement de confiance peuvent avoir une influence sur les performances du système, nous proposons dans cette section une évaluation théorique sur les coûts introduits par le mécanisme de sécurité proposé.

5.1 Coût en volume de données supplémentaire

Nous envisageons l'utilisation de la cryptographie symétrique pour ses plus faibles coûts, et plus particulièrement de keyed-Hash Message Authentication Code (HMAC) pour assurer l'authenticité et l'intégrité des messages échangés entre les TH. La taille de l'en-tête de sécurité ajouté à tout message peut être considérée comme constante et dépendante de trois paramètres : la taille du condensé, la taille du numéro de séquence, et la taille de l'identifiant de l'émetteur. Si nous prenons l'hypothèse d'un numéro de séquence et d'un identifiant chacun représenté sur 16 bits, puis d'un condensé de 160 bits (pour la fonction de hachage SHA-1), la taille de l'en-tête de sécurité est alors de 192 bits par message, et 208 bits par message ACK. Par ailleurs, le volume supplémentaire d'octets associé aux messages ACK est linéaire avec le nombre de messages et de voisins.

Notons que tout système assurant l'authenticité et l'intégrité des messages induit un coût dû au temps de calcul pour la génération du condensé. Sur une carte à puce, ce coût peut être considéré comme relativement faible.

5.2 Coût en nombre de messages supplémentaire

Tout message émis par un nœud source est immédiatement acquitté par chacun des nœuds présents dans son voisinage direct de communication, et ce message ACK valide la réception d'un seul message. Cette approche a pour conséquence une augmentation importante du trafic en nombre de messages. Néanmoins, d'autres schémas moins coûteux en nombre de messages

additionnels peuvent être envisagés. Par exemple, plutôt que d'exiger un ACK pour chaque message, une solution pour réduire le nombre de messages ACK transmis consisterait pour un nœud à accumuler les accusés de plusieurs messages reçus (en provenance de plusieurs sources), puis de les envoyer dans un seul message ACK.

5.3 Coût en espace mémoire requis

Chaque nœud émetteur maintient une table contenant le numéro de séquence et le temps d'expiration associé de tous les messages pour lesquels il est en attente d'un acquittement. Ici, le temps d'expiration représente le temps pendant lequel un message peut être acquitté par un nœud voisin. Une fois ce temps dépassé, l'entrée est supprimée de la table. En prenant l'hypothèse qu'une entrée de cette table est représentée sur 48 bits (16+32), si une nœud émet localement M messages, alors il en résulte une occupation mémoire de 48*M bits. Soit pour un scénario où un nœud émet $M = 10$ messages par seconde, l'occupation mémoire est de 60 octets.

Une autre table contenant l'ensemble des identifiants de nœuds présents dans son voisinage et leur degré de confiance (calculé à partir des messages acquittés) est également maintenue (soit par exemple 16+16=32 bits par entrée). L'occupation mémoire de cette table est linéaire avec le nombre de nœuds voisins. Si chaque nœud a en moyenne $N = 50$ voisins, il en résulte une occupation mémoire de 200 octets.

Les cartes à puce actuelles sont dotées de 64 à 256 ko de mémoire vive, ce qui couvre les besoins en espace mémoire requis par notre application. Notons par ailleurs que les cartes à puce de nouvelle génération offrent de plus grandes capacités de stockage et peuvent par conséquent être déployées pour répondre à des scénarii plus complexes (faisant interagir plus de nœuds, et impliquant plus de messages).

6 Travaux antérieurs

Dans le contexte des réseaux mobiles ad hoc, un protocole de routage sécurisé au moyen d'un environnement matériel de confiance a été proposé par [4]. Alors que cette approche offre des garanties de sécurité au niveau transport, elle est fondée sur une hypothèse forte qui est l'existence d'une chaîne de confiance entre le processus de routage, le pilote de la carte réseau sans-fil et la carte réseau sans-fil. En effet, avant que toute communication ne soit engagée, le processus de routage requiert une attestation certifiant que le pilote de la carte réseau remplira ses obligations (à savoir auditer puis reporter les performances de fonctionnement de la carte réseau sans-fil au processus de routage).

Plusieurs propositions fondées sur un système de monnaie virtuelle [2,3,1] modélisent le problème de la retransmission des messages comme étant un marché économique dans lequel les opérations d'émission et de retransmission ont un coût. Buttyán et Hubaux ont proposé un mécanisme distribué de rétribution basé sur l'échange d'une monnaie virtuelle appelée Nuglets [1]. Le principe de fonctionnement est le suivant : chaque nœud maintient un compteur de crédit ; lorsqu'un nœud émet un message, il perd des crédits correspondant au coût de retransmission produit par les nœuds intermédiaires impliqués dans la transaction ; et inversement, lorsqu'un nœud retransmet un message, il accumule des crédits. Pour qu'un nœud puisse émettre ses propres messages sur le réseau, son compteur de crédit doit être positif. Ainsi, cette approche vise à niveau applicatif à stimuler les nœuds à participer aux opérations d'acheminement des messages de données. Pour assurer la sécurité du mécanisme de rétribution, soit éviter toute manipulation frauduleuse du compteur de crédits, chaque nœud est

muni d'un environnement sécurisé à l'intérieur duquel sont stockés les outils cryptographiques (paire de clés publique/privée du module de sécurité, clés de session) utilisés pour authentifier les messages, un compteur de crédits local, et où sont d'autre part exécutées les opérations de paiement.

Contrairement à ces travaux, nous avons d'une part étudié les problèmes existants lorsque aucune confiance explicite n'est accordée dans l'environnement matériel et logiciel de l'hôte, puis nous avons proposé un protocole d'identification du comportement des hôtes dans les couches basses de communication pouvant être utilisé à plus haut niveau dans un système de récompense ou de réputation.

7 Conclusion

Dans cet article, nous avons étudié le problème de la sécurisation des protocoles distribués grâce au support d'équipements matériels de confiance. Nous avons montré que du fait de la non-fiabilité des canaux physiques de communication entre un TH et son hôte, il était impossible de garantir le fonctionnement correct des protocoles distribués uniquement en sécurisant leur exécution sur chacun des nœuds participants.

Nous avons proposé un protocole d'identification par le matériel de confiance du comportement des hôtes au regard des opérations bas niveau d'émission et de réception des messages. L'analyse réalisée fait ressortir des possibles mesures exploitables par un TH pour rendre plus fiable les protocoles distribués. Nous avons observé qu'à cause d'attaques par émission ciblée, nous sommes limités à évaluer la qualité des liens entre les hôtes. Même si ces informations sont beaucoup moins précises que ce que nous aurions pu espérer avoir, nous considérons qu'elles peuvent servir à la prise de décision pour des systèmes de plus haut niveau.

Références

1. Levente Buttyán and Jean-Pierre Hubaux. Stimulating cooperation in self-organizing mobile ad hoc networks. *MONET*, 8(5) :579–592, 2003.
2. Jean-Pierre Hubaux, Thomas Gross, Jean-Yves Le Boudec, and Martin Vetterli. Towards self-organized mobile ad hoc networks : The terminodes project. *IEEE Communications Magazine*, 39(1) :118–124, 2001.
3. Markus Jakobsson, J. P. Hubaux, and L. Buttyan. A Micro-Payment Scheme Encouraging Collaboration in Multi-Hop Cellular Networks. In *Financial Crypto 2003*, 2003.
4. Michael Jarrett and Paul Ward. Trusted computing for protecting ad-hoc routing. In *CNSR*, pages 61–68. IEEE Computer Society, 2006.
5. Chai-Keong Toh. A novel distributed routing protocol to support ad hoc mobile computing. In *Proc. IEEE 15th Annual International Phoenix Conference on Computers and Communications, IEEE IPCCC 1996, March 27-29, Phoenix, AZ, USA*, pages 480–486. IEEE, IEEE, March 1996.

Quelle confiance dans les composants matériels ?

Loïc Duflot

Direction centrale de la sécurité des systèmes d'information
51 boulevard de La Tour-Maubourg
75700 Paris cedex SP

Résumé Nous présentons ici quelques unes des problématiques relatives à la sécurité des composants matériels. Nous montrons en particulier comment l'absence de modèle de sécurité au niveau matériel, la complexification des architectures matérielles et la multiplication des nouvelles technologies au sein des processeurs et des chipsets peuvent avoir un impact sur la sécurité d'une plateforme de confiance. Nous présentons par exemple comment un attaquant peut contourner toute fonction de sécurité implémentée au niveau logiciel en exploitant un bogue ou un piège d'un processeur. L'étude se concentre sur les microprocesseurs x86 (32 ou 64 bits) et les architectures matérielles associées.
Mots-clés : processeur, chipset, piégeage matériel, bogue matériel, x86.

1 Introduction

L'informatique de confiance repose sur différentes briques logicielles et matérielles à partir desquelles la confiance dans une plateforme peut être obtenue, presque par transitivité. Jusqu'à il y a quelques années, la communauté scientifique s'est essentiellement penchée sur la question de la confiance dans les composants logiciels. Différentes questions ont été posées telles que celles de savoir comment développer un composant logiciel sans bogue ou quel doit être le périmètre de tel ou tel composant logiciel. Ce n'est que très récemment que l'intérêt de la communauté pour les questions de confiance dans le matériel est apparu. Au delà de la confiance dans le Trusted Platform Module [25], c'est bien de la confiance dans les processeurs et les chipsets qu'il s'agit. Comment avoir confiance dans un système logiciel quel qu'il soit si les composants matériels mis en œuvre par la plateforme sont piégés ou bogués ? Nous allons ici nous intéresser à cette problématique. Nous allons montrer comment l'absence de modèle de sécurité au niveau matériel a déjà par le passé pu être exploitable par d'éventuels attaquants, et comment la complexification des architectures matérielles rend de plus en plus difficile la formalisation d'un modèle de fonctionnement du matériel. Ensuite, nous étudierons les conséquences sur la sécurité des systèmes d'exploitation ou des moniteurs de machine virtuelles d'un bogue ou d'un piégeage dans un composant.
Nous avons fait le choix de concentrer notre étude sur les processeurs de la famille x86 (32 bits ou 64 bits), et sur les chipsets associés. Les processeurs Pentium®, Xeon®, Core Duo^{TM}, Athlon^{TM}, Turion^{TM} font partie de cette famille.

2 Quelques éléments d'architecture

La figure 1 présente l'architecture classique d'une plateforme disposant d'un processeur x86. Dans le principe (on verra plus loin que cette affirmation est aujourd'hui largement erronée), la majeure partie des composants logiciels s'exécute sur le processeur : code du système

d'exploitation, des applications, des éventuels moniteurs de machines virtuelles, code de maintenance s'exécutant en mode "System Management" par exemple. Dans leur mode nominal de fonctionnement (mode protégé pour les processeurs 32 bits ou IA32-e pour les processeurs 64 bits), les processeurs disposent d'un mécanisme dit de privilège processeur (encore appelé CPL ou ring [19]). Les tâches les plus privilégiées (le système d'exploitation ou le moniteur de machine virtuelle selon les architectures logicielles) s'exécutent en ring 0 et disposent des privilèges maximaux au niveau matériel. Les applications utilisateur s'exécutent elles en ring 3 et ont donc des privilèges très restreints. Ce mécanisme de ring couplé aux mécanismes de segmentation et de pagination mis en œuvre au sein de l'unité de gestion de la mémoire du processeur (MMU) permet à un système d'exploitation d'isoler son propre espace mémoire de celui des autres applications potentiellement malicieuses, mais également d'isoler le contexte d'une application donnée de celui des autres applications.

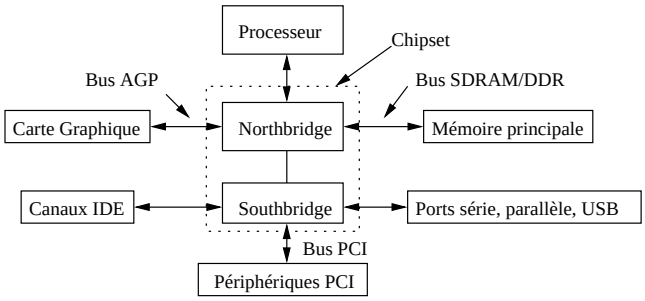


FIG. 1: Architecture simplifiée d'un système x86 (détail : Pentium® 4)

Les processeurs les plus récents munis des technologies AMD-V et VT-x [20] disposent de plus d'un mode de fonctionnement particulier permettant, lorsqu'un moniteur de machines virtuelles est mis en œuvre, d'isoler les composants de confiance des autres composants (systèmes d'exploitation invités par exemple).
Les processeurs x86 disposent enfin d'un mode de fonctionnement très privilégié appelé System Management Mode (SMM) [20] dans lequel vont s'exécuter un certain nombre de routines de traitement d'événements liés à la gestion d'alimentation de la plateforme indépendamment du système d'exploitation. Des exemples de tels événements peuvent être, en fonction des systèmes : connexion d'un réseau, passage à l'an 2000, connexion d'un périphérique USB, mise en veille de la machine. L'emploi du mode System Management permet au concepteur de la machine de spécifier le comportement attendu du matériel en fonction des différents événements pouvant se produire, sans préjuger du système d'exploitation qui sera ensuite mis en œuvre.

3 Absence de modèle de sécurité matériel

Ce n'est que très tardivement que de réelles considérations de sécurité ont été prises en compte dans l'étude du mode de fonctionnement des composants matériels. Certes, ce sont bien des considérations de sécurité qui ont conduit à l'implémentation de mécanismes tels que le mécanisme de ring, de segmentation ou l'ajout de flag NX [4] ou XD [18] au mécanisme

de pagination. Si donc les processeurs ont rapidement fourni aux systèmes d'exploitation modernes des mécanismes leur permettant a priori d'assurer leur sécurité au niveau logique, aucune étude de cohérence au niveau de la plateforme n'a jamais été menée. Des mécanismes de sécurité ont été petit à petit rajoutés à la demande au sein de différents composants sans qu'une étude de cohérence de l'ensemble n'ait été menée. Il n'y a donc aucune raison particulière pour que l'ensemble soit cohérent du point de vue de la sécurité.

L'exemple le plus patent est sans doute l'incohérence engendrée par le mécanisme dit "d'accès direct à la mémoire" (DMA). Pour des raisons de performances, il est possible pour les périphériques d'accéder directement à la mémoire sans aucun contrôle du processeur. Alors que les applications subissent un contrôle permanent effectué par le système d'exploitation, les périphériques eux, pouvaient historiquement lire ou modifier à volonté le contenu de l'ensemble de l'espace mémoire de la machine. Dès lors deux pistes d'attaques étaient possible pour tout attaquant souhaitant par exemple modifier le noyau du système d'exploitation ou retrouver des clefs de chiffrement stockées en mémoire :

- connecter un périphérique à la machine et initier depuis ce périphérique des accès DMA [14] ;
- reconfigurer logiquement, et sans accès physique à la machine, un contrôleur du chipset (par exemple le contrôleur USB [1]) pour que ce dernier réalise pour son compte des accès directs en mémoire.

Ces deux pistes d'attaque ont été prouvées exploitables. Les dernières générations de chipsets mettent cependant en œuvre des technologies dites VT-d ou I/OMMU [3] permettant de limiter les zones mémoire auxquels les périphériques ont accès via le mécanisme DMA. Cette réponse n'est que partielle, d'une part dans la mesure où ces technologies sont à l'heure actuelle assez peu utilisées, en partie car certains développeurs de périphériques et de pilotes ont pris de mauvaises habitudes qui empêchent leurs périphériques de fonctionner si la protection au niveau des accès DMA est activée, et d'autre part dans la mesure où cette réponse ne résout que le problème des accès DMA. Il a été prouvé à plusieurs reprises que des mécanismes de traduction d'adresse du chipset [1,27] pouvaient être exploités par un attaquant à des fins d'escalade de privilèges.

De même, le modèle de transition entre modes des processeurs a été assez peu étudié, et il a également été montré [16] que sous certaines conditions un attaquant pouvait générer des transitions non contrôlées vers le mode System Management lui permettant d'exécuter des commandes dans ce mode très privilégié et donc d'obtenir les privilèges maximums sur le système. Cette technique a depuis été reprise pour permettre à des virus de type rootkit de cacher certaines de leurs fonctions [5,17].

4 Conséquences de l'augmentation du nombre des fonctionnalités

Compte-tenu de ce qui précède, il aurait été raisonnable, pour les constructeurs de machines et de composants, de tenter d'établir un modèle de sécurité global de la machine de manière à corriger les problèmes de sécurité résiduels. Au lieu de cela, la tendance est aujourd'hui à la multiplication des fonctionnalités. Les processeurs sont aujourd'hui tous multi-cœurs, comportent des fonctionnalités cryptographiques avancées, supportent jusqu'à 5 modes de fonctionnement complètement différents et gèrent des registres spécifiques (en particulier les registres de type MSR Model Specific Registers [20]) différents d'une version à l'autre et pouvant avoir un impact significatif sur le sécurité de l'ensemble.

Pire, la complexité des chipsets croît de jour en jour. Un chipset actuel comporte bien entendu des contrôleurs mémoire et des contrôleurs de bus, mais aussi des contrôleurs de périphé-

riques intégrés (carte réseau, contrôleur graphique) et plusieurs microprocesseurs génériques programmables. Ces derniers peuvent par exemple être utilisés pour émuler un composant de type TPM (Trusted Platform Module), effectuer des vérifications d'intégrité en mémoire (utilitaire Intel DeepWatch [6] présenté lors de la conférence Blackhat 2008), ou encore permettre l'administration du chipset à distance. C'est par exemple l'objet de la technologie Intel AMT [11] visant à permettre une administration d'un parc de machines à distance depuis un centre d'administration. Le centre d'administration se connecte directement à l'environnement local d'administration qui s'exécute entièrement dans le chipset (environnement de type web à base de technologies commes et SOAP, qui ne sont pas nécessairement réputées pour leur fiabilité). Des technologies similaires existent pour la remontée d'alarme (technologie ASF [12] par exemple), qui permettent au chipset de prendre la main sur la carte réseau et de diffuser des messages d'alertes.

Globalement la vision de l'architecture présentée sur la figure 1 tend à disparaître pour être remplacée par une architecture où un seul composant, le chipset, gère la quasi totalité des périphériques, implémente des mécanismes de sécurité et de cloisonnement, et des fonctions d'administration et exécute des programmes logiciels très complexes, qui n'ont pas de raison d'être plus sécurisés que ceux qui s'exécutent sur le processeur, le tout sans modèle de sécurité apparent. Le processeur n'est plus vu que comme un coprocesseur du chipset qui gère les applications utilisateur, et sur lequel le chipset a un droit de regard et de modification. Ce changement de philosophie explique l'intérêt grandissant de la communauté pour les questions de sécurité liées au chipset.

5 Conséquences d'un bogue ou d'un piégeage d'un processeur

Au vu de ce qui précède, il doit sembler évident que tout bogue ou piégeage d'un chipset peut avoir des conséquences dramatiques sur la sécurité de l'ensemble. Dans cette partie, nous nous concentrons maintenant sur l'étude des conséquences d'un bogue ou d'un piégeage des processeurs sur l'efficacité des mécanismes de sécurité matériels.

Lors de la conférence d'introduction aux journées C&ESAR 2007 [7], Adi Shamir a présenté l'impact sur la sécurité de l'implémentation logicielle de certains systèmes de chiffrement asymétrique, d'un bogue ou d'un piégeage de l'unité arithmétique et logique d'un microprocesseur x86 [19]. Cette présentation a ensuite été effectuée lors de la conférence Crytpo 2008. La presse grand public s'en est largement faite écho [24].

D'autres présentations ont récemment montré comment il était possible d'implémenter simplement des pièges exploitables dans les processeurs. On peut notamment citer la présentation de l'université de l'Illinois [22] qui a montré comment piéger de différentes façons un microcontrôleur de type Sparc et exploiter les pièges implémentés. Une autre présentation [13] a étudié l'ajout à un processeur ARM d'un mode semblable au mode SMM du Pentium permettant d'obtenir les privilèges maximums sur une machine.

5.1 Bogue, piégeage ou fonction non documentée ?

Les termes de bogue, piégeage ou fonctionnalité non documentée renvoient à trois notions intuitivement différentes. Un bogue correspond à une erreur non volontaire d'implémentation d'un composant qui se traduit par un dysfonctionnement dudit composant, incompatible avec les spécifications de ce dernier. La présence de bogues dans un composant matériel semble inévitable avec les méthodologies de développement actuelles, étant donnée la relative difficulté à corriger a posteriori un bogue matériel, malgré le soin extrême généralement apporté par les concepteurs.

On parle, en revanche, de fonctionnalité non documentée, lorsque le développeur a inclus dans son produit un ensemble de fonctionnalités dont il n'a pas précisé l'existence, la syntaxe ou la sémantique. Il est très fréquent pour un concepteur d'implémenter des fonctionnalités non documentées, en particulier pour ce qui est des fonctions de débogage ou d'administration. Il est courant, même si c'est intuitivement incorrect, de considérer que le secret de l'existence d'un mécanisme empêchera toute exploitation à des fins frauduleuses de ce dernier. À titre d'exemples, plusieurs fonctionnalités des processeurs x86 n'ont pas, à ce jour, été documentées. Les processeurs x86 possédaient en particulier une instruction assembleur dite LOADALL[1] [8] qui permettait au code qui l'exécutait de charger en un cycle d'horloge l'ensemble des registres du processeur à partir d'une image en mémoire dont l'adresse était passée en paramètre à l'instruction. On imagine sans peine l'intérêt qu'une telle fonctionnalité peut avoir dans le domaine de l'analyse de fonctionnement du processeur ou du débogage, mais également quel usage un attaquant pourrait faire de cette fonctionnalité. Les processeurs x86 actuels possèdent également quelques fonctionnalités non documentées par les constructeurs, parmi lesquelles l'instruction assembleur "salc" et dont on trouve la signification par ailleurs [9].

Enfin, la notion de piégeage renvoie, elle, à une volonté manifeste, de la part du développeur ou d'une entité étant parvenu d'une manière ou d'une autre à s'immiscer dans le flot de conception du composant cible, d'introduire des fonctionnalités non documentées qui lui permettront a posteriori, lorsque le composant sera produit et distribué, d'effectuer des opérations les plus privilégiées possibles à l'insu de l'utilisateur légitime. On peut par exemple imaginer l'exemple paranoïaque d'un piégeage d'une carte réseau qui sur réception d'une trame IP particulière, passerait dans un mode actif permettant des accès arbitraires à distance en mémoire principale via le mécanisme DMA (Direct Memory Access [14]). Un autre exemple classique est celui d'une carte à puce piégée qui lorsqu'elle reçoit une donnée x, renvoie systématiquement le chiffré de x par une clef K sauf pour une valeur donnée de x pour laquelle elle renvoie K.

Bien que ces trois notions renvoient à des concepts différents l'on est malheureusement contraint, du point de vue d'une analyse sécurité, de considérer leur équivalence. En effet, un expert en sécurité qui devrait évaluer le niveau de sécurité d'un composant n'aurait connaissance ni des bogues spécifiques à ce composants (non connus du développeur), ni aux fonctions non documentées, et encore moins aux éventuels piégeages. Le principe de précaution lui impose en outre de considérer l'impact maximal de chacun des problèmes potentiels. Bien que dans la plupart des cas, les bogues puissent être anodins et non exploitables par un quelconque attaquant, dans le pire cas, un bogue peut être exploitable et permettre à l'attaquant de mettre en œuvre une escalade de privilèges sur le système. Il en va de même des fonctionnalités non documentées. Du point de vue d'une analyse sécurité, ces trois notions sont donc équivalentes.

Nous utiliserons donc dans la suite de ce document le terme piégeage pour désigner indifféremment un bogue, une fonction non documentée ou un piégeage en tant que tel, dès lors qu'ils sont exploitables par un attaquant.

5.2 Expérimentations

Afin d'étudier les conséquences d'un bogue ou d'un piégeage sur la sécurité des systèmes d'exploitation et des moniteurs de machines virtuelles, deux expérimentations ont été menées. La première (figure 2a) consistait à implémenter dans un processeur x86 un piège qui, lorsque le processeur se trouvait dans un état donné et exécutait une instruction particulière, donnait

[1] Cette instruction assembleur était toutefois a priori réservée au seul usage des composants s'exécutant en ring 0.

à la tâche courante les privilèges maximums (ring 0) sur le système. La seconde (figure 2b) implémentait un piégeage un peu plus compliqué qui fournissait en outre un moyen de contourner les mécanismes de segmentation et de pagination lorsque le piège était activé.

Pour le besoin des preuves de concept, l'émulateur libre Qemu a été modifié pour implémenter les pièges considérés.

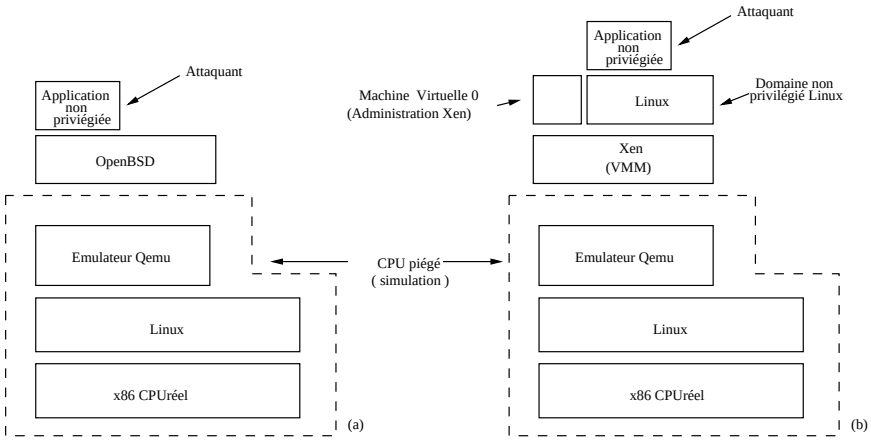


FIG. 2: Schéma du dispositif expérimental

5.3 Conclusions de l'étude

Il est apparu lors de l'étude (voir [15] pour une description plus détaillée) que le premier piégeage (figure 2a) permettait à un attaquant en ayant connaissance d'obtenir l'exécution de code arbitraire en couche noyau quel que soit le système d'exploitation mis en œuvre, depuis des privilèges très restreints (contexte d'une application non privilégiée). En revanche, il est apparu très difficile (aucun moyen satisfaisant n'a été mis en évidence jusqu'ici) d'exploiter un tel piégeage dans le cas général si un moniteur de machines virtuelles (Xen [26] par exemple) est utilisé. Le second piégeage (figure 2b) permet à l'attaquant de prendre la main sur n'importe quel système (exécuter du code en ring 0), qu'il mette en œuvre un moniteur de machines virtuelles ou non, quels que soient les mécanismes de sécurité mis en œuvre, et ce depuis des privilèges très réduits (application non privilégiée d'un système d'exploitation lui-même non privilégié).

Cette étude montre qu'il sera toujours possible à un attaquant d'intégrer dans un processeur un piégeage simple et générique qui lui permette a posteriori de prendre la main sur tous les systèmes qui mettent en œuvre ce composant dès lors qu'il est capable d'exécuter du code sur ce dernier. Cela prouve une fois encore le risque qu'il existe à faire tourner une application non maîtrisée sur une machine même lorsque celle-ci ne bénéficie initialement que de privilèges très restreints.

5.4 Limiter l'impact des bogues et des piégeages

L'une des questions qui s'est rapidement posée est celle de la détection de tels piégeages. Une équipe de recherche a présenté [2] comment des techniques proches de celles mises en œuvre dans les attaques par canaux auxiliaires [23] pouvaient permettre de détecter des pièges intégrés en phase de fonderie dans les composants. L'efficacité pratique d'une telle méthode reste cependant à vérifier. Il est en particulier de notoriété publique que l'administration américaine, via l'agence DARPA (Defense Advanced Research Projects Agency), a lancé un grand projet de 3 ans (projet "Trust In Circuits") visant à déterminer la meilleure méthode pour détecter un piège au sein d'un composant.

Afin de limiter le risque d'exploitation d'un bogue ou d'un piège par un attaquant, il convient dans la mesure du possible que les concepteurs de système n'autorisent que des composants logiciels de confiance à s'exécuter sur la machine. Tout code non maîtrisé s'exécutant sur une machine a la possibilité d'exploiter d'éventuels bogues ou d'éventuels piégeages pour obtenir les privilèges maximaux sur une machine. La suppression des moyens de compilation ou d'exécution de code arbitraire sur une machine (macros) est également une bonne pratique.

5.5 Aspects pratiques

Il est intéressant de noter que les deux constructeurs principaux de processeurs x86 (Intel et AMD) publient régulièrement une liste [10] des bogues matériels de leurs processeurs. Cette liste est généralement relativement longue et certains bogues qui y sont répertoriés ne seront sans doute jamais corrigés, du fait de la difficulté de modifier a posteriori le fonctionnement d'un circuit microélectronique aussi complexe qu'un microprocesseur.

Un chercheur indépendant (Kris Kaspersky) a prévu de présenter [21] comment exploiter un certain nombre de ces bogues comme moyen d'escalade de privilège. Selon lui certains de ces bogues sont même exploitables à distance et ce quel que soit le système d'exploitation mis en œuvre. Ces allégations restent à l'heure où ces lignes sont écrites à vérifier mais la situation n'en est pas moins préoccupante.

6 Conclusion

En conclusion, nous avons ici abordé la question de la confiance dans les composants informatiques standards que sont les processeurs et les chipsets. Nous avons montré que cette confiance, pourtant nécessairement préalable à la mise en œuvre de tout système d'information sécurisé, était pourtant dans l'absolu difficilement atteignable. Les publications académiques et scientifiques qui démontrent les faiblesses des architectures matérielles actuelles sont de plus en plus nombreuses et pertinentes. Face à ce constat, il est important de prendre en compte d'éventuels bogues ou piégeages des composants matériels lors de l'analyse de risque préalable à la conception de tout système destiné à la protection d'information sensibles.

Références

1. L. Absil and L. Duflot. Programmed i/o accesses : a threat to virtual machine monitors. In *Pacific security conference PacSec07*, 2007.
2. D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi, and B. Sunar. Trojan detection using ic fingerprinting. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 296–310, 2007.

3. AMD. I/o virtualization technology (iommu) specification. 2006. http://www.amd.com/us-en/assets/content_type/white_papers_and_tech_docs/34434.pdf.
4. Advanced Micro Devices (AMD). Nx flag by amd. 0. http://www.amd.com.
5. BSDDaemon, Coideloko, and D0nAnd0n. System management mode hack : Using smm for other purposes. In *Phrack Magazine*, 2008. http://www.phrack.org/issues.html?issue=65\&id=7\#article.
6. Y. Bulygin. Insane detection of insane rootkits : Chipset-based approach to detect virtualization. In *Blackhat Briefings USA*, 2008.
7. CELAR. Computer and electronics security applications rendez-vous (C&ESAR 2007). 2007. http://www.cesar-conference.fr/.
8. R. Collins. The loadall instruction. In *Tech Specialist Journal*, 1991. http://www.x86.org/articles/loadall/tspec_a3_doc.htm.
9. R. Collins. Undocumented opcodes : Salc. 1999. http://www.rcollins.org/secrets/opcodes/SALC.html.
10. Intel Corp. Intel core 2 extreme processor x6800 and intel core 2 duo desktop processor e6000 and e4000 sequence : Specification update. 2007. http://www.intel.com/technology/architecture-silicon/intel64/index.htm.
11. Intel Corp. Active management technology : Open source drivers and tools. 2008. http://www.openamt.org.
12. Intel Corp. Alert standard format : Standards-based systems management. 2008. http://download.intel.com/design/network/papers/ASF_whitepaper.pdf.
13. F. David, E. Chan, J. Carlyle, and R. Campbell. Cloaker : Hardware supported rootkit concealment. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 296–310, 2008.
14. M. Dornseif. Owned by an ipod : Firewire/1394 issues. In *CanSecWest security conference core05*, 2005. http://cansecwest.com/core05/2005-firewire-cansecwest.pdf.
15. L. Duflot. Cpu bugs, cpu backdoors and consequences on security. In *ESORICS 2008 : Proceedings of the 13th European Symposium on Research Computer Security*, 2008.
16. L. Duflot, D. Etiemble, and O. Grumelard. Security issues related to pentium system management mode. In *Cansecwest security conference Core06*, 2006. http://www.cansecwest.com/slides06/csw06-duflot.ppt.
17. Shawn Embleton and Sherri Sparks. The system management mode (smm) rootkit. In *Black hat briefings*, 2008.
18. Intel Corp. Execute disable bit software developer's guide. 0. http://cache-www.intel.com/cd/00/00/14/93/149307_149307.pdf.
19. Intel Corp. Intel 64 and ia 32 architectures software developer's manual volume 1 : basic architecture. 2007. http://www.intel.com/design/processor/manuals/253665.pdf.
20. Intel Corp. Intel 64 and ia 32 architectures software developer's manual volume 3b : system programming guide part 2. 2007. http://www.intel.com/design/processor/manuals/253669.pdf.
21. K. Kaspersky. Remote code execution through intel cpu bugs. In *Hack In The Box Security Conference*, 2008.
22. S. King, J. Tucek, A. Cozzie, C. Grier, W. Jiang, and Y. Zhou. Designing and implementing malicious hardware. In *Proceedings of the first usenix workshop on large scale exploits and emergent threats, LEET'08*, 2008.

23. P. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In *CRYPTO'99 : Proceedings of Advances in Crytology*, 1999.
24. H. Morin. Deux experts mettent en garde sur la sécurité sur internet. In *Le Monde*, 2007.
25. Trusted Computing Group. Tpm specification version 1.2 : Design principles. 2008. `http://www.trustedcomputinggroup.org/specs/TPM/MainP1DPrev103.zip`.
26. University of Cambridge. Xen virtual machine monitor. 2007. `http://www.cl.cam.ac.uk/research/srg/netos/xen/documentation.html`.
27. Rafal Wojtczuk. Subverting the xen hypervisor. In *Blackhat Briefings USA*, 2008.

Un panorama des techniques de furtivité et de leur détection

Jean-Marie Fraygefond, Jean-Marie Borello, Didier Eymery

CELAR, BP 7419,
35174 BRUZ Cedex, France
{jean-marie.fraygefond, jean-marie.borello, didier.eymery}
@dga.defense.gouv.fr

Résumé Ces dernières années, de nouvelles fonctionnalités de virtualisation ont été intégrées aux processeurs (architectures Intel® VT-x et AMD-V™) afin de simplifier et d'améliorer les performances des machines virtuelles.

Ces fonctionnalités ont aussi fait l'objet d'une attention toute particulière en terme de codes malveillants afin de tirer profit des propriétés d'isolation offertes par la virtualisation. En effet, ces architectures ont été conçues pour que théoriquement, un système d'exploitation ne puisse détecter s'il est ou non virtualisé. C'est ainsi qu'en 2006 sont apparus les premiers prototypes d'hyperviseurs malveillants (désignés sous le nom de *Virtual Machine Monitor* VMM) permettant de virtualiser un système d'exploitation sur des architectures le supportant.

Cet article présente les différentes techniques de furtivité employées jusqu'alors ainsi que les techniques de détection qui leur sont associées.
Mots-clés : furtivité, rootkit, code malveillant, détection de malwares.

1 Introduction

Parmi les différentes menaces pesant sur un système d'information, celle représentée par les codes malveillants est incontournable. Dans le cadre des architectures de confiance, l'une des questions fondamentales consiste à déterminer si une machine donnée est compromise ou non. D'un côté, les attaquants cherchent à pénétrer un système et à s'y maintenir le plus longtemps possible. De l'autre côté, les défenseurs cherchent quant à eux à déceler le moindre signe de compromission ou d'activité illicite sur une machine.

Afin de dissimuler leur présence au sein d'un système, les codes malveillants (malware) emploient des techniques dites de furtivité. Cette catégorie de codes malveillants est communément désignée sous le terme de "rootkits". Leur but consiste généralement à cacher des ressources particulières (processus, fichiers, clés de registre, ports ouverts, etc) ou encore à fournir un accès à une "backdoor" [11].

Avec l'arrivée des processeurs supportant la virtualisation, un nouveau type de rootkits a vu le jour. Ces derniers se comportent alors comme des hyperviseurs virtualisant à la volée un système d'exploitation afin de dissimuler leur présence.

Ce article se compose de deux parties : une première partie présente les différentes techniques de furtivité en fonction de leur degré d'interaction avec le système d'exploitation hôte. Une deuxième partie expose les approches de détection pour chaque technique de furtivité.

2 Techniques de furtivité

Historiquement les techniques employées par les rootkits étaient classées en deux catégories distinctes suivant leur niveau d'action : mode utilisateur (ring3) ou mode noyau (ring0) [6]. Depuis quelques années, une nouvelle catégorie de rootkit exploitant les capacités de virtualisation des nouveaux processeurs a vu le jour. Joanna Rutkowska [23] fut la première à proposer une nouvelle taxonomie permettant d'inclure cette nouvelle catégorie de rootkits désignée sous le terme le terme de *Hypervisor Virtual Machine* (HVM) rootkit . Cette classification est fondée sur l'interaction du programme malveillant avec le système d'exploitation.

2.1 Codes malveillants de type 0 : interactions "documentées" avec le système

Cette catégorie comprend les programmes utilisant uniquement des interfaces documentées du système. Il s'agit des toutes premières techniques de camouflage historiquement utilisées qui tirent profit des possibilités offertes par le système d'exploitation. La figure 1 illustre de tels programmes. Généralement, il ne s'agit pas de furtivité au sens propre du terme mais plutôt

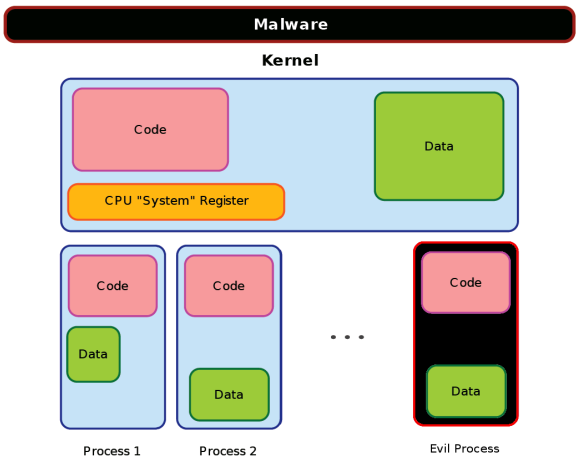


FIG. 1: Code malveillant de type 0 [23].

de se soustraire aux outils de base fournis avec le système. Ces techniques étant largement documentées [6], nous les rappelons brièvement :

– **Des propriétés spécifiques au système d'exploitation.** Par exemple, l'explorateur de Windows, par défaut, n'affiche pas les fichiers possédant les attributs "hidden" ou "system".

Un exemple connu de malware utilisant cette propriété à des fins de camouflage est le ver Nimda [26] qui restaurait l'option de masquage des fichiers cachés afin de dissimuler sa présence.

- **L'installation d'un rootkit en tant que service.** Une autre technique simple de dissimulation consiste à lancer un processus en tant que service afin de le dissimuler. Le rootkit doit cependant posséder des privilèges administrateurs pour pouvoir créer et lancer un service.
- **L'installation d'un rootkit en tant que driver.** Une technique très employée consiste à lancer un driver permettant de masquer des informations à l'utilisateur tel que des fichiers, des clés de registres, etc. Comme dans le cas d'un service, le rootkit doit posséder des privilèges administrateurs pour pouvoir créer et lancer un driver. L'exemple le plus célèbre est le rootkit de SONY (voir 2.2).

Un exemple célèbre directement en rapport avec la thématique des architectures de confiance, est celui de Thompson [25]. Dans cet article, l'auteur s'interroge sur la confiance que l'on peut accorder à un fichier source dans le cas où l'outil de compilation n'est pas lui même entièrement maîtrisé. Plus précisément, il est remarqué qu'un compilateur C est lui même écrit en langage C. Aussi, si le premier compilateur (servant à compiler le second) contient un virus qui se propage dans le binaire produit lors de la compilation, alors le second compilateur sera lui même infecté. Ainsi, tout binaire produit par ce compilateur sera infecté bien que les sources du second compilateur soient totalement saines. Autrement dit, comme tout binaire provient d'un binaire (le compilateur) et de sources, il est illusoire de reposer sa confiance sur le code source uniquement.

2.2 Codes malveillants de type 1 : modification des constantes du système

Cette catégorie de malware modifie des valeurs normalement constantes au sein du système, qu'il s'agisse de code[1] ou encore de données, aussi bien dans des exécutables en mode utilisateur qu'en mode noyau (voir figure 2).

Les techniques employées sont dites d'interception ("hooking") aussi bien en mode utilisateur qu'en mode noyau. La figure 3 montre le flux d'exécution lors d'un appel à la fonction `FindNextFile` de la part d'un processus en mode utilisateur. Les numéros représentent les emplacements où s'appliquent les différentes techniques de "hooking". Certaines techniques présentées ci-après, ne sont plus utilisables aujourd'hui notamment avec la venue de PatchGuard [15] voir 3.3 qui interdit certaines modifications au niveau du noyau.

Les "hooks" en mode utilisateur Toutes ces techniques sont employées en mode utilisateur, c'est à dire en (1) sur la figure 3.

- **Injection de code** : L'API Win32 fournit un ensemble de fonctionnalités permettant la création et l'exécution de code au sein d'un autre processus. Pour ce faire, la première étape consiste à allouer dynamiquement de la mémoire dans l'espace d'adressage d'un processus en cours d'exécution au moyen de la fonction `VirtualAllocEx()`. Le code à exécuter est ensuite copié dans ce nouvel espace mémoire via la fonction `WriteProcessMemory()`. Enfin, un thread est créé à l'aide de la fonction `CreateRemoteThread()` afin d'exécuter le code précédemment copié.

[1] On suppose ici que le code n'est pas auto modifiant. Bien qu'il s'agisse d'une entorse aux possibilités des processeurs actuels, cette hypothèse s'avère correcte dans la majorité des cas correspondant à des applications légitimes (autres que malveillantes).

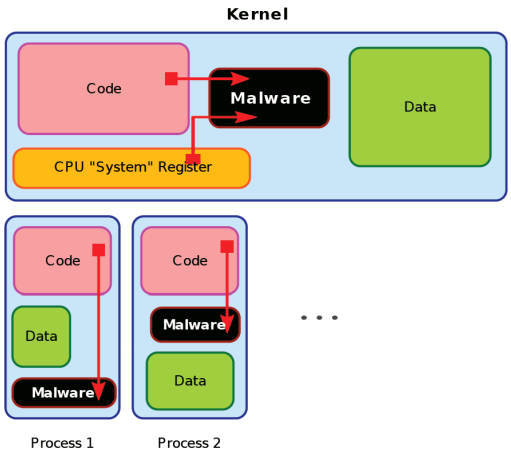


FIG. 2: Code malveillant de type 1 [23].

- **Injection d'une bibliothèque dynamique (DLL)** : La technique d'injection de code précédemment décrite est difficile puisque la totalité du code injecté doit être relative c'est à dire indépendante de l'adresse à laquelle est copiée le code en question. Un moyen efficace de contourner cette difficulté est de charger une DLL qui contient l'ensemble du code à exécuter par le thread. Par défaut tout processus possède la fonction `LoadLibrary()` permettant le chargement dynamique d'une bibliothèque. Le plus simple est de faire pointer l'adresse de départ du thread sur la fonction `LoadLibrary()` à laquelle est passé en paramètre le chemin de la DLL à exécuter. Lors de la création du thread, la DLL est chargée. Juste après le chargement, la fonction `DllMain()` est automatiquement appelée ce qui permet l'exécution du code.
- **Détournement d'une interface de programmation (API)** : Il s'agit de modifier le flux d'exécution d'un processus lors d'un appel particulier à une API afin de lui faire exécuter le code désiré. La figure 4 présente un exemple de détournement de l'appel `CreateFileA` de la DLL `kernel32`. Le bloc (1) montre les quatre premières instructions d'origine de la fonction. Le bloc (2) présente le code de la même fonction une fois détourné : les trois premières instructions ont été changées par un appel à la fonction dite trampoline représentée en bloc 3. Ce bloc 3 constitue le code que l'on veut dissimuler. Après exécution de ce bloc, les trois premières instructions d'origine de la fonction `CreateFileA` sont exécutées (bloc 3) avant de retrouver le flux d'exécution légitime (bloc 2).

Les "hooks" en mode noyau Toutes ces techniques cherchent à détourner des fonctionnalités du noyau. Les emplacement mentionnés correspondent aux numéros (2), (3), (4) et (5) sur la figure 3.

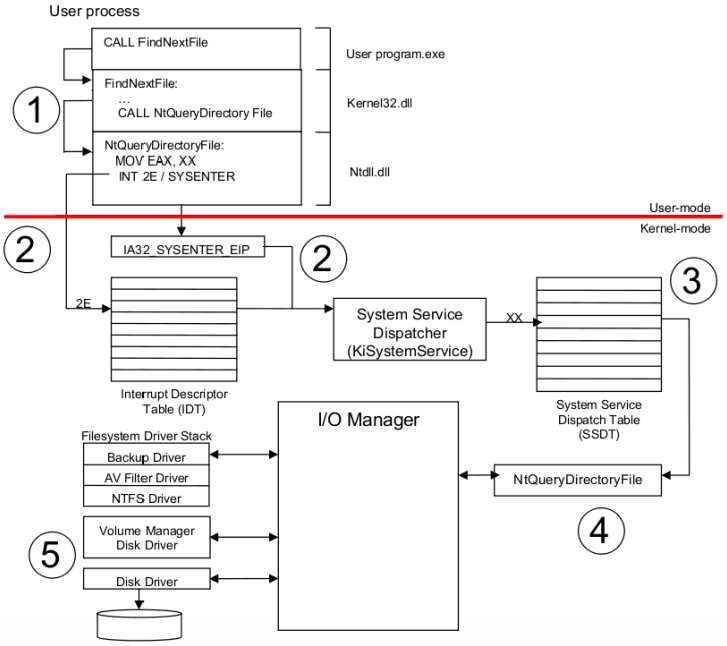


FIG. 3: Différentes possibilités de hooking.

- **Patch de l'Interrupt Descriptor Table (IDT), des instructions SYSENTER et syscall(2).** L'IDT est une table d'au maximum 256 entrées stockant les adresses des procédures qui gèrent les interruptions matérielles ou logicielles. En patchant l'IDT, il est possible de détourner les interruptions afin de faire exécuter discrètement du code. Notamment l'interruption 0x2E sous Windows NT et 2000 qui sert aux appels systèmes (voir `NTDLL.dll`). Pour Windows XP et Vista 32bits, les appels systèmes se font via l'instruction `SYSENTER`. Trois registres particuliers (MSR) spécifient l'adresse et le pointeur de pile pour cette instruction. Pour Windows Vista x64, le passage en mode noyau se fait via l'instruction `syscall` qui utilise les *System-Linkage Registers* (`STAR`, `LSTAR`, `CSTAR`).
- **Patch de la System Service Descriptor Table (SSDT) (3).** La *System Service Descriptor Table* (SSDT) est la table des services offerts par le noyau pour les applications en mode utilisateur.
- **Patch dynamique du noyau (4) ou de périphériques (drivers) (5).** Après la SSDT, un rootkit peut descendre à un niveau encore plus bas afin de modifier directement le code

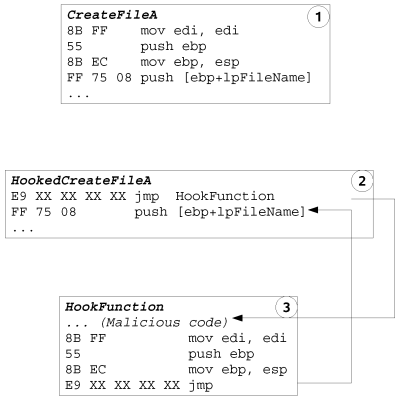


FIG. 4: Exemple de détournement de la fonction `CreateFileA`.

des services système au sein du noyau. Il peut de même modifier un driver précis afin de le détourner.

Étude de cas : le rootkit de SONY L'exemple de rootkit de type 1 le plus connu est celui de SONY découvert par Mark Russinovich [21]. Cet exemple permet de récapituler les principales techniques employées par des rootkits de type 1 sur un cas réel largement médiatisé.

Afin de se prémunir contre la copie illicite d'un de ces CD, SONY a fourni un logiciel spécifique de DRM (*Digital Rights Management*) pour écouter et copier (seulement 3 fois) les morceaux de musique contenus dans ce CD. Ce logiciel comprenait notamment un driver `Aries.sys` modifiant la SSDT afin de dissimuler les fichiers, répertoires, clés de registres et processus dont le nom commence par `$sys$`. La figure 5 présente deux captures d'écrans illustrant la modification de la SSDT. La première capture (à gauche) montre le code du driver permettant de modifier certaines entrées de la SSDT. La deuxième capture (à droite) montre le contenue de la SSDT. On constate que tous les services légitimes exportés par cette table pointent vers la même plage d'adresse dans l'espace noyau (adresses comprises entre 0x80501030 et environ 0x80640000) sauf les deux pointeurs encadrés en rouge qui correspondent aux deux fonctions détournées. Ce driver à vocation de DRM présentait toutes les caractéristiques d'un véritable rootkit. Pire, un logiciel malveillant aurait pu profiter de ce rootkit afin de s'installer dans un répertoire non visible et de s'exécuter de manière furtive (avec un nom commençant par `$sys$`).

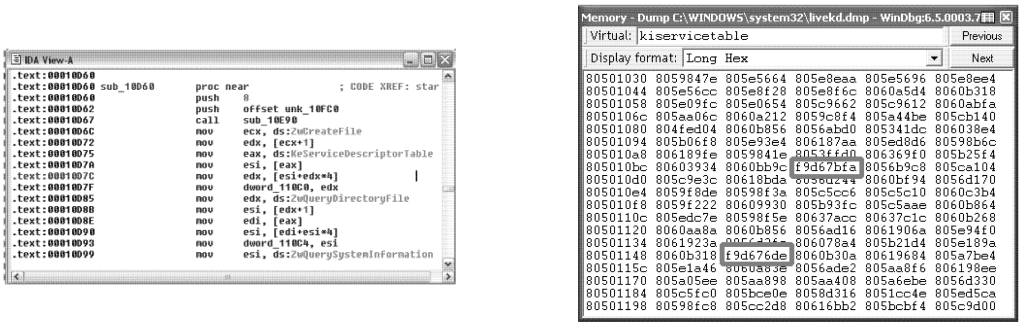


FIG. 5: Exemple de modifications de la SSDT extrait de [21] (code du driver à gauche et résultat à droite).

2.3 Codes malveillants de type 2 : modification des variables du système

Contrairement à la catégorie précédente qui modifie les données constantes du système, les malwares de type 2 modifient les données variables comme l'indique la figure 6.

Au niveau du noyau ces techniques sont connues sous le terme de *Direct Kernel Object Modification* (DKOM). Une technique couramment employée est celle illustrée par le rootkit FU [8] qui consiste à cacher certains processus particuliers en modifiant la liste des processus actifs du système. On retrouve aussi ce procédé dans le ver `W32/Fanbot.A` [7]. Cette technique est illustrée à travers la figure 7. Elle consiste à récupérer la structure `ETHREAD` du thread dont on veut masquer le processus. Un champ particulier correspond à un pointeur sur une structure `EPROCESS` représentant notre processus dans la liste des processus actifs. Chaque structure `EPROCESS` possède deux pointeurs vers des structures similaires représentant le processus précédent (pointeur `BLINK`) et le processus suivant (pointeur `FLINK`). Reste alors à modifier ces pointeurs (`BLINK` et `FLINK`) pour le processus précédent ainsi que pour le processus suivant afin de faire disparaître notre processus.

2.4 Codes malveillants de type 3 : virtualisation du système

L'étape suivante dans l'évolution des rootkits conduit logiquement à des programmes ne modifiant pas le système d'exploitation (voir figure 8). Cette étape, qui peut sembler surprenante à priori, a été abordée par le rootkit Microsoft SubVirt [13]. Ce rootkit persistant modifiait l'amorce de la machine afin de pouvoir lancer le système d'exploitation d'origine à l'intérieur d'une machine virtuelle.

L'apparition de la virtualisation matérielle au sein de nouveaux processeurs (AMD-V [2] et Intel VT-x [12]) a permis une nouvelle évolution vers des rootkits non persistant se comportant comme des hyperviseurs vis à vis du système d'exploitation. Deux prototypes visant chacun une architecture ont été développés : BluePill [22] sur AMD-V et Vitriol [28] sur Intel VT-x.

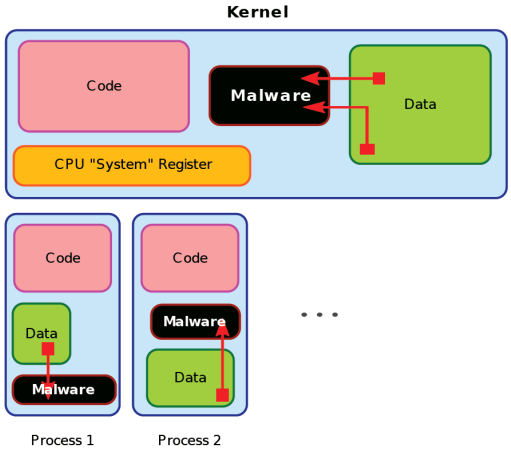


FIG. 6: Code malveillant de type 2 [23].

Contrairement à tous les précédents rootkits présentés, un rootkit HVM est fondamentalement différent car il n'a aucun point d'attache avec le système d'exploitation virtualisé. En effet, du fait de la conception de ces nouveaux processeurs, il n'y a aucun moyen direct pour un système d'exploitation de savoir s'il est virtualisé ou non.

Virtualisation d'un système d'exploitation Un hyperviseur traditionnel ou virtualiseur (VMWare, VirtualPC, VirtualBox,...) définit une machine virtuelle, lance le système d'exploitation et intercepte chaque accès au matériel. Un hyperviseur "matériel" définit l'environnement du système virtualisé grâce a des structures de contrôle particulières (VMCS pour Intel et VMCB pour AMD) qui décrivent l'état du processeur, les contrôles d'entrée et de sortie définissant les transitions entre l'hyperviseur et le système virtualisé, ainsi que les instructions à intercepter (voir [2] pages 369-370).

La virtualisation d'un système nécessite les étapes suivantes :

1. **Vérifier le support de la virtualisation (SVM/VMX).**
   Cette vérification est faite via l'instruction `CPUID`. Pour AMD, la fonction `CPUID` 0x80000001 doit retourner le bit SVM (Secure Virtual Machine) dans le registrer `ECX`. Pour Intel, la fonction `CPUID` 0 doit retourner le bit VMX (Virtual Machine Extension) dans le registre `ECX`. Ensuite, l'option de virtualisation doit être activée. Pour AMD, affecter le bit `SVME` du registre `EFER`. Pour Intel, affecter le bit `VMXE` du registre `CR4`.
2. **Allouer et initialiser les structures VMCB / VMCS.**
   L'instruction permettant de basculer en mode hyperviseur prend en paramètre l'adresse physique d'une page de 4Ko contenant la *Virtual Machine Control Block / Structure* (VMCB / VMCS). Cette structure contient :

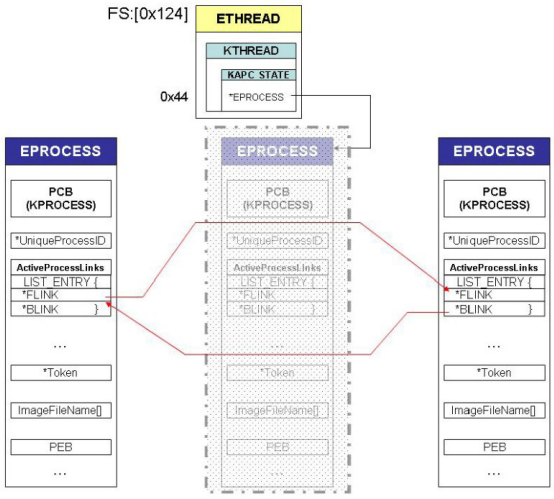


FIG. 7: Illustration de la liste des processus actifs (*EPROCESS*).

- une liste d'instructions et d'évènements devant être interceptés ainsi que les routines de traitement des ces interceptions ;
- des valeurs de contrôles spécifiant l'environnement d'exécution de la machine virtuelle ainsi que les actions à réaliser avant de continuer l'exécution du système virtualisé ;
- l'état du processeur "invité".

3. **Basculer en mode hyperviseur.**
   Une fois toutes les étapes préalables accomplies, l'hyperviseur exécute l'instruction de virtualisation du système (pour AMD `VMRUN`, et pour Intel `VMLAUNCH`).

Étude de cas : BluePill BluePill est un prototype de rootkit HVM dont le fonctionnement est illustré sur la figure 9.

Pour commencer BluePill alloue un bloc de mémoire (au sens OS du terme) afin d'y dupliquer la table des pages en cours. Cette copie sera celle de l'hyperviseur, l'originale étant dédiée a l'OS virtualisé. Cependant, tout se passe comme si l'hyperviseur et l'OS virtualisé possèdaient la même table des pages.

Ensuite, pour chaque processeur, BluePill duplique l'IDT (sauf pour le vecteur de faute de protection générale) et recrée une copie quasiment complète de la GDT. Il alloue alors une page (4KB) pour le VMCB (état de l'OS virtualisé et liste des évènements que l'hyperviseur souhaite voir notifiés) et le HSA (zone de données gérée par l'UC et contenant l'état de l'hyperviseur au moment où il exécute un "guest"). Les seuls évènements que BluePill souhaite

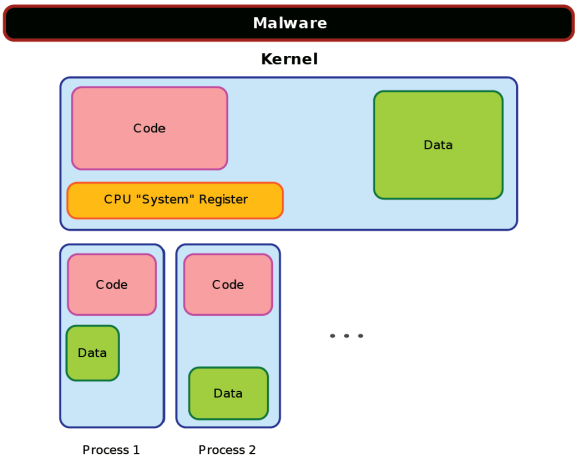


FIG. 8: Code malveillant de type 3.

voir sont ceux concernant l'utilisation des instructions liées a la virtualisation (principalement instructions VMxxx), plus l'instruction CPUID (pour le knock). Autrement dit, BluePill laissera l'OS "virtualisé" vivre sa vie sans interférer en aucune façon avec lui. L'OS aura un accès plein et entier à tout le matériel et à toute la mémoire...
Lorsque tout est fin prêt, BluePill active le mode hyperviseur, active la table des pages dupliquées (registre CR3). À partir de là l'UC va entrer dans une boucle sans fin consistant a exécuter la machine virtuelle (instruction VMRUN) et à traiter les évènements en sa provenance (ceux déclarés dans le VMCB). Il faut noter que l'instruction VMRUN est bloquante. Elle ne termine que lorsqu'un évènement, associé à la machine virtuelle, est actif. BluePill commence l'exécution de l'OS virtualisé là où a commencé la virtualisation. Du point de vue de l'OS rien ne s'est passé.

BluePill est donc un démonstrateur de mise en œuvre des instructions de virtualisation des nouveaux processeurs. Ce n'est pas un malware dans la mesure où il n'interagit pas avec l'invité. Pour un malware, les informations manipulées par l'invité sont sa raison d'être.

3 Techniques de détection

«That which cannot be prevented should be detected; that which cannot be detected should be prevented.»

La détection d'un rootkit installé sur un système d'exploitation avec des outils d'investigation s'appuyant sur l'OS est théoriquement impossible puisque le rootkit "est" l'OS et qu'il peut

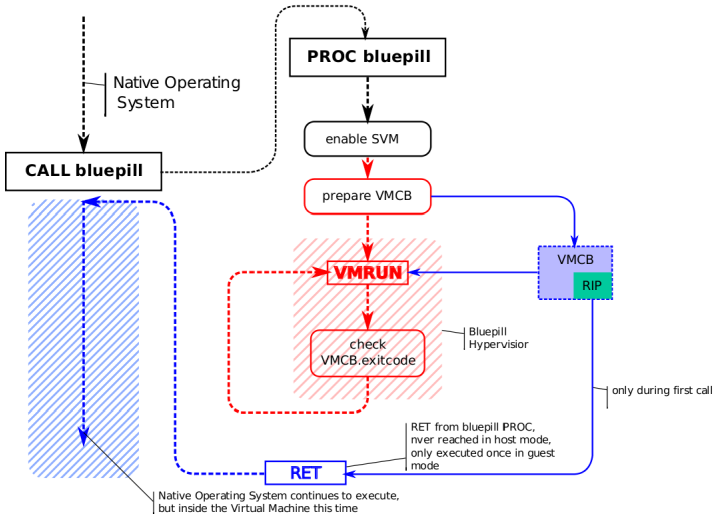


FIG. 9: Illustration de la virtualisation d'un système d'exploitation [24].

réaliser tous les filtrages nécessaires à sa furtivité. De plus la détection à posteriori repose sur l'interprétation des données retournées par les outils d'investigation. N'ayant pas forcément connaissance du fonctionnement interne de ces outils, il peut être difficile :
- de comprendre et interpréter les résultats fournis ;
- de comprendre et interpréter l'absence de résultats ;
- tout simplement de leur faire confiance !

Cependant nous allons voir qu'il existe un fossé entre la théorie et la pratique du fait de la complexité des OS actuels et des actions à prendre en charge par le rootkit pour s'assurer une invisibilité totale. Nous assistons à l'éternelle lutte acharnée que se livrent d'un côté les programmeurs de malwares ayant une connaissance de plus en plus pointue des systèmes d'exploitation ainsi qu'une imagination sans limite, et de l'autre, les éditeurs de solutions de sécurité qui cherchent des parades à ces attaques.

Les techniques de détection mises en œuvre dans les outils de défense ne sont pas ou peu documentées afin de retarder au maximum les attaquants dans la mise en œuvre de techniques anti-forensic. Plusieurs approches sont utilisées pour détecter un rootkit, basées sur une détection de violation d'intégrité de l'OS :
- le contrôle d'intégrité ;
- la détection d'incohérences (vues croisées) ;
- la détection par signatures.

Le contrôle d'intégrité, tout comme la recherche de signatures visent à détecter la présence de rootkits et de leurs modules. L'approche par vues croisées vise la détection des effets du rootkit, c'est-à-dire les objets cachés par le rootkit (fichiers, clés de registre, processus, ports...) ou encore des techniques de détection pro-actives. Toutes ces techniques étant bien évidemment complémentaires.

3.1 Détection des codes malveillants de type 0 : interactions "documentées" avec le système

Comme vu précédemment les rootkits de cette catégorie exploitent les lacunes de l'OS en terme de reporting (par exemple, le gestionnaire de tâches `taskmgr.exe` de Windows est simpliste, il n'affiche pas les processus qui s'exécutent en tant que services) ou des paramètres de configuration par défaut altérant la perception de l'utilisateur (non affichage des extensions connues par exemple, utilisation de caractères spéciaux dans les noms de fichiers, attributs "hidden" des fichiers) ou tout simplement des fonctionnalités de l'OS (Alternative Data Streams (ADS) du système de fichier NTFS). La détection de tels rootkits est triviale pour un utilisateur averti c'est à dire ayant une bonne connaissance du système et des outils de reporting disponibles (process explorer, tasklist /svc).

3.2 Codes malveillants de type 1 : modification des constantes du système

Face à cette catégorie de codes malveillants, les principales techniques de détection reposent sur des mécanismes de contrôle d'intégrité.

Secure File Check (SFC) et Windows File Protection (WFP) Ces applications sont deux fonctionnalités complémentaires apparues avec les versions de Windows 2000 pour répondre en partie à un problème récurrent : le "DLL Hell problem". Sur les versions de l'OS Microsoft antérieures à windows 2000 n'importe qu'elle application disposant des droits d'administration pouvait remplacer ou supprimer des modules du système d'exploitation et notamment l'API win32.

Ces modifications étaient un facteur majeur d'instabilité de l'OS et les programmeurs de malware ont bien sûr exploité cette possibilité pour installer des rootkits, soit en remplaçant certains modules de l'OS par des versions modifiées ou en installant des "DLL proxy" agissant comme un filtre vers les APIs cibles [18]

Il est à noter que la vulnérabilité exploitée est double :

- mauvais comportement des utilisateurs qui exécutent des programmes en tant qu'administrateur et ne respectent pas le principe du moindre privilège ;
- méconnaissance des éditeurs de logiciels qui introduisent des vulnérabilités et encouragent le comportement cité ci-dessus.

La réponse de Microsoft à ce problème fut l'implémentation d'une technique de détection pro-active dans son système d'exploitation. WFP est en fait un démon qui surveille en permanence certains fichiers et répertoires systèmes, empêchant ainsi leur modification ou leur suppression. Microsoft n'a pas documenté ce mécanisme qui a fait l'objet de plusieurs études et de nombreuses attaques [4] permettant de le désactiver temporairement ou définitivement. Dès lors que le système possède une fonctionnalité de mise à jour (Windows Update) il existe forcément un mécanisme interne permettant de s'affranchir de cette protection, puisque l'OS doit être en mesure de modifier des composants systèmes.

`SFC.exe` quant à lui est un utilitaire qui permet de scanner "à la demande" les fichiers protégés par WFP pour détecter d'éventuelles violation d'intégrité et procéder au remplacement des versions altérées à partir d'une source sûre : CD/DVD rom d'installation de Windows ou un répertoire cache contenant les versions intègres des modules.

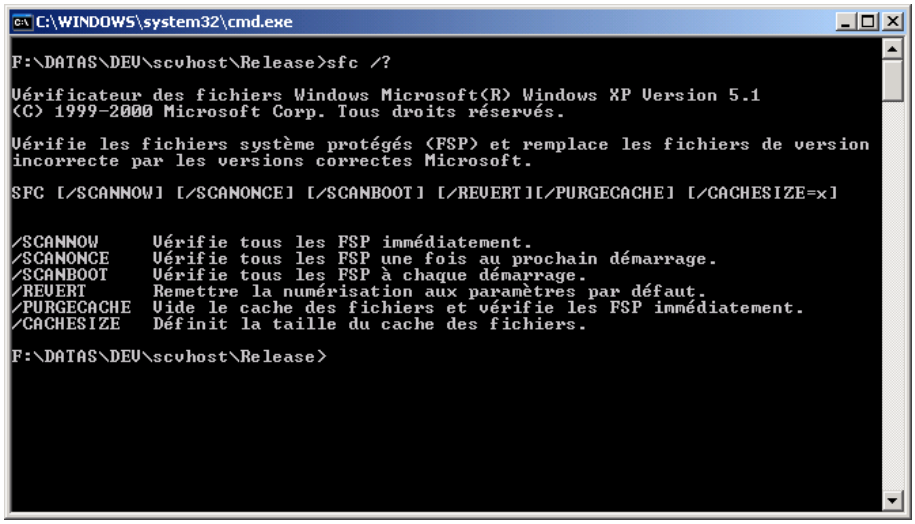


FIG. 10: sfc.exe

EyeBootRoot En août 2005, des consultants en sécurité de la société eEye ont présenté à la conférence BlackHat un prototype de rootkit nommé EyeBootRoot [5] qui introduisait une porte dérobée dans le système Windows. La spécificité de ce rootkit vient du fait qu'il s'installe en mémoire lors de la phase d'amorce de l'OS. Cette phase d'amorce fait intervenir plusieurs composants logiciels :

1. Le BIOS

2. Le Master Boot Record
3. Le secteur de boot
4. Le loader
5. et enfin le noyau de l'OS

Sur une plateforme Intel, le microprocesseur est en mode réel lors de la phase d'amorce. Dans ce mode réel, un programme peut accéder à l'ensemble du jeu d'instructions ainsi qu'à toute la mémoire sans contraintes de sécurité. Sous Windows, c'est le loader qui va commuter le microprocesseur en mode protégé après avoir créé les structures de données nécessaires (IDT, GDT). EyeBootRoot tire partie de cette vulnérabilité en s'intercalant entre le BIOS et le Master Boot Record. 2 ans après la présentation de EyeBootRoot, les premières implémentations malveillantes de cette preuve de concept étaient repérées dans la nature [9,14]. EyeBootRoot pourrait en première analyse s'apparenter à un virus de boot classique, cependant EyeBootRoot possède quelques spécifi-cités :

- contrôle total du démarrage de l'OS, survit notamment au passage du mode réel au mode protégé ;
- aucun fichier support, le rootkit peut résider dans des secteurs disques ;
- aucune entrée dans la base de registre (survit au reboot par modification du Master Boot Record).

L'éradication de ce rootkit est relativement simple puisqu'il suffit de restaurer une constante système : le MBR d'origine, mais sa détection est difficile car l'empreinte du rootkit sur le système est très faible.

La solution proposée par Microsoft à ce type d'attaque se nomme **BitLocker**. Apparue avec Windows Vista, BitLocker est une solution de chiffrement intégral de la partition système au niveau secteur pouvant s'appuyer sur une puce TPM (*Trusted Platform Module*) pour assurer un contrôle d'intégrité de la séquence d'amorce. La séquence d'amorce peut être vue comme une chaîne de confiance. Chaque constituant de la chaîne "mesure" le constituant suivant (calcule un Hash cryptographique) et stocke le résultat dans le TPM avant de donner la main. La figure 11 montre un détail de l'implémentation de cette technique dans le MBR de Vista. Au final chaque élément de la chaîne est intègre car il a été mesuré et chargé par un constituant intègre. La racine de cette chaîne de confiance est le Core Root of Trust for Measurement (CRTM) situé dans le BIOS (compatible TPM) et supposée immuable.

3.3 Codes malveillants de type 2 : modification des variables du système

Face aux techniques de furtivité de plus en plus évoluées employées par les Rootkits (hooking, DKOM), les défenseurs ont du réagir et revoir leurs techniques de détection. Une nouvelle approche consiste, non pas à tenter de détecter une violation d'intégrité de l'OS mais plutôt tenter de trouver ce qui est caché par le système à la manière de la lampe à UV utilisée par les services de police scientifique. C'est ce que tentent de faire `BlackLight` (F-Secure) et `RootKitRevealer` (Microsoft), `PatchGuard` quant à elle, est une technique de détection pro-active.

PatchGuard (aka Kernel Patch Protection) `PatchGuard` est une fonctionnalité introduite en 2005 avec les versions 64 bits de windows XP et windows Server 2003 SP1. C'est la réponse de Microsoft aux tentatives de modifications d'intégrité du noyau par les développeurs indépendants de logiciels. Plus généralement KPP ne vise pas uniquement les RootKits de type 2 (hooks SSDT) mais toutes les applications tierces qui injectent du code dans le noyau (Drivers) et réalisent des modifications à la volée. Les anti-virus, firewalls, HIPS

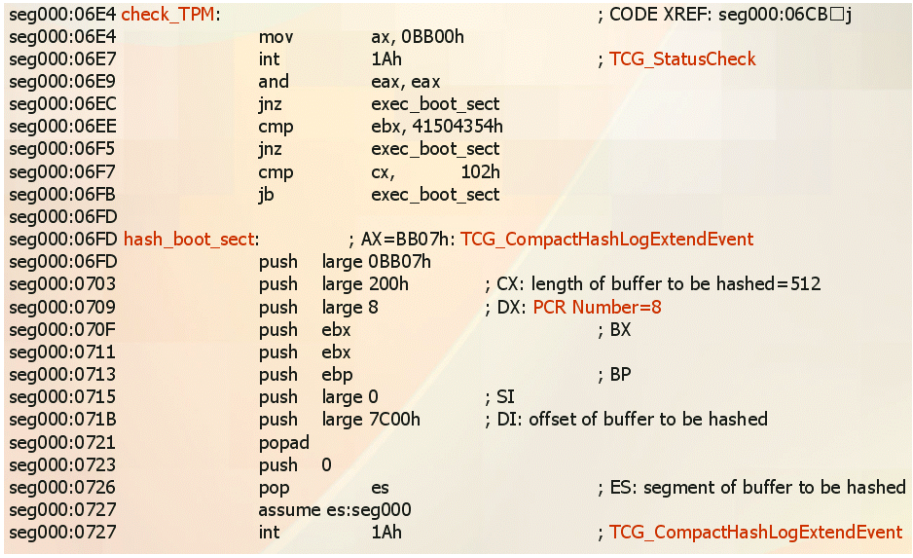


FIG. 11: Master Boot Record Vista

et autres produits de "sécurité" sont eux-même très friands de ces techniques de hot-patching. Or le noyau Windows est une structure opaque, non documentée et susceptible de changer à tout moment. L'introduction de modifications dans le noyau par des applications tierces est une source potentielle d'instabilité et de contre-performance, c'est ce que Microsoft essaye d'empêcher.

`PatchGuard` protège en intégrité des structures internes résidant en mémoire : IDT, GDT, SSDT, Images système (ntoskrnl.exe, ndis.sys, hal...), MSRs (syscall). Cependant cette protection repose sur un principe de "sécurité par l'obscurité" (obfuscation de code, symboles absents) et `PatchGuard` a déjà connu deux mises à jour importantes suite aux attaques et techniques de contournement développées [17].

`PatchGuard` surveille les structures internes citées ci-dessus et en cas de violation d'intégrité la protection lève une exception et arrête le système. Cette fonctionnalité a été vivement critiquée par les éditeurs d'antivirus dont les produits réalisaient eux-même des patchs dans le noyau.

BlackLight [1] et RootKitRevealer [3] ces deux outils tentent une approche par vues croisées. L'objectif est de révéler des fichiers et/ou clés de registre cachés en mettant en évidence des incohérences en comparant les résultats fournis lors de l'appel aux APIs Windows et les résultats obtenus en analysant directement les structures de bas niveau sur disque.

Les APIs Win32 ont pour objectif de masquer au programmeur la complexité des objets manipulés. En effet pour lire une fichier sur disque un programme en mode utilisateur va utiliser l'API Win32 `ReadFile()` qui elle même s'appuie sur l'API Native NtReadFile qui elle même fait appel à des composants noyau (pilote du système de fichier, pilote de bus, pilote périphérique...) pour atteindre le matériel.

Le programmeur manipule une structure logique (un HANDLE) sans se soucier de la représentation sur disque de cette structure (système de fichier FAT, NTFS...,) ni des aspects liés au matériel (géométrie du disque). Mais le programmeur peut aussi décider d'aller lire le fichier directement sur les secteurs du disque en réimplémentant son propre pilote. C'est ce que font BlackLight et RootKitRevealer.

Tout écart constaté entre les deux techniques peut être révélateur d'un filtre placé quelque part dans le système. La figure 12 schématise le fonctionnement de RootKitRevealer

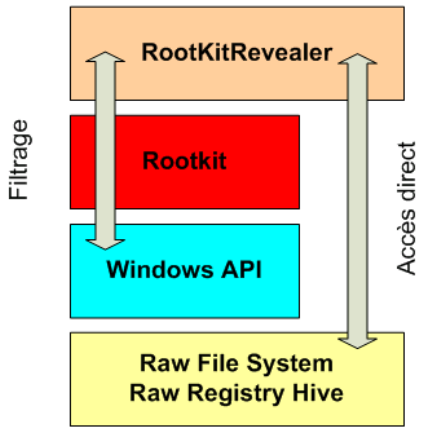


FIG. 12: RootkitRevealer

Pour les processus et les threads camouflés via des techniques comme DKOM, de manipulation directe des objets du noyau, BlackLight fait un appel à l'API native `NtOpenProcess` pour tous les PIDs possibles, essayant ainsi d'obtenir éventuellement un HANDLE valide sur un processus non visible via l'API :
`CreateToolHelp32Snapshot` [27]. Il est difficile pour un rootkit de leurrer un outil comme RootKitRevealer. Il doit pour cela modifier les appels aux APIs de haut niveau, avoir une connaissance très fine des structures de données sous jacentes (système de fichier, base de registre) et une connaissance approfondie du fonctionnement interne de l'outil de détection.

À notre connaissance, un tel niveau de sophistication n'a encore jamais été rencontré dans un rootkit. Cependant il existe une vulnérabilité inhérente à tous les outils de détection reposant sur ce mécanisme de différenciation. Une contre-attaque possible pour le rootkit consiste tout

simplement à désactiver ses filtres s'il détecte la présence d'un outil comme BlackLight. Dans ce cas, il n'y a plus d'écart entre un appel haut niveau et un appel bas niveau. Pour contreparer, BlackLight et RootKitRevealer utilisent des techniques anti-debug pour camoufler leur présence et éviter leur détection par les rootkits (une inversion des rôles en quelque sorte!). BlackLight notamment injecte sa librairie d'accès bas niveau dans le shell graphique de windows (processus `explorer.exe`). Il devient ainsi compliqué pour le rootkit de prendre une décision : s'agit-il d'un accès de BlackLight ou bien d'une opération d'un utilisateur?

Il est à noter que les rootkits qui opèrent sans hooking, c'est-à-dire en modifiant directement les structures de données internes de l'OS (comme FU), ne tentent pas de se camoufler eux-mêmes et ne sont pas capables de camoufler des fichiers ou des entrées de la base de registre. À ce titre, ils sont toujours détectables avec les techniques classiques de détection par signature implémentées dans les anti-virus.

3.4 Codes malveillants de type 3, BluePill et les hyperviseurs

Avec l'apparition de Bluepill, le problème de la détection a pris une autre dimension puisque le rootkit n'est plus dans l'OS mais à l'extérieur de l'OS. Agissant comme un hyperviseur, il peut accéder à l'OS mais l'inverse n'est pas vrai [10]. La détection de BluePill depuis l'OS virtualisé est théoriquement impossible si la virtualisation est complètement transparente. De nombreuses discussions ont eu lieu autour de la virtualisation hardware et de son usage dans un cadre malveillant [16]. En fait il y a souvent eu confusion entre deux objectifs complètements différents : détecter un rootkit, c'est à dire un code malveillant et détecter la virtualisation. La détection de virtualisation, qu'elle soit implémentée de façon matérielle ou logicielle (VmWare, VirtualPC, VirtualBox...), dépend fortement de la capacité de l'architecture sous-jacente à supporter la virtualisation d'un point de vue théorique, or il a été démontré [20] que l'architecture Intel ne remplissait pas les conditions énoncées en 1974 par Popek et Goldberg [19] pour supporter une virtualisation complètement transparente. Dans le cas de BluePill, et plus généralement des hyperviseurs malveillants, on peut essayer de faire abstraction de l'architecture matérielle et admettre le fait qu'ils sont indétectables. Dans ce cas on revient à une problématique de détection de canaux auxiliaires, et de fuite d'information. À savoir qu'il sera toujours possible d'essayer de détecter le rootkit par ses effets induits, notamment au niveau des entrées/sorties matérielles (réseaux...). Une vulnérabilité des hyperviseurs concerne la persistance. L'hyperviseur doit survivre aux redémarrages de la machine et donc il doit "exister" physiquement sur un média. Il devient donc détectable par les techniques classiques de recherche par signatures.

3.5 Conclusion

Au moment où vous lisez ce texte, d'autres techniques de furtivité et de détection sont développées. Dans le domaine de la lutte informatique, l'attaquant dispose de la position la plus confortable, en ayant toujours un coup d'avance. Le simple fait de publier les détails d'une technique de détection lui procure un avantage, momentané, puisqu'il peut implémenter la contre-mesure adaptée. Cependant les techniques présentées dans ce papier sont, pour certaines, "extrêmes" dans le sens où elles sont hors de portée technique de la plupart des auteurs de code malveillants et que vous ne les rencontrerez probablement jamais. La majorité des rootkits publics implémentent des techniques de hooking aisément détectables avec les produits de sécurité du marché. Qu'en est-il du rootkit complètement furtif, indétectable, quel que soit l'outil utilisé ou la compétence de l'analyste? Soit ce monstre n'existe pas, soit il existe, et personne ne l'a jamais rencontré puisqu'il est indétectable. Ce qui nous ramène à

une question d'ordre plus philosophique : le monde perçu est-il le monde réel? Qu'est-ce qui est réel finalement ? Et on revient à la question fondamentale en sécurité : où plaçons-nous notre confiance ?

Références

1. F-secure blacklight. http://www.f-secure.com/blacklight/.
2. Amd64 architecture programmer's manual ; volume 2 : System programming, September 2007. http://www.amd.com/us-en/assets/content_type/white_papers_and_tech_docs/24593.pdf.
3. Bryce Cogswell and Mark Russinovich. Rootkitrevealer v1.71, 2006. http://technet.microsoft.com/fr-fr/sysinternals/bb897445(en-us).aspx.
4. Jeremy Collake. Hacking windows file protection. http://www.bitsum.com/aboutwfp.asp.
5. eEye Digital Security Research. Bootroot. http://research.eeye.com/html/tools/RT20060801-7.html.
6. Gergely Erdélyi. Hide'n'seek ? anatomy of stealth malware. In *Proceedings of Virus Bulletin Conference 2003*, 2003.
7. Description of w32/fanbot.a@mm. http://securityresponse.symantec.com/avcenter/venc/data/w32.fanbot.a@mm.html.
8. Fu rootkit. http://www.rootkit.com/project.php?id=12.
9. Stealth mbr rootkit, 2008. http://www2.gmer.net/mbr/.
10. Matt Hines. Showdown at the blue pill corral, 2007. http://securitywatch.eweek.com/showdown_at_the_blue_pill_corral.html.
11. G. Hoglund and J. Butler. *Rootkits : Subverting the Windows Kernel.* Addison-Wesley, 2005.
12. Intel 64 and ia-32 architectures software developer's manual ; volume3b : System programming guide, part 2, November 2007. http://download.intel.com/design/processor/manuals/253669.pdf.
13. Samuel T. King, Yi-Min Wang, Peter M. Chen, Chad Verbowski, Helen J. Wang, and Jacob R. Lorch. Subvirt : Implementing malware with virtual machines. In *Proceedings of the 2006 IEEE Symposium on Security and Privacy*, 2006. http://www.eecs.umich.edu/~pmchen/papers/king06.pdf.
14. Brian Krebs. New nasty hides from windows, anti-virus tools, 2008. http://blog.washingtonpost.com/securityfix/2008/01/new_nasty_hides_from_windows_a.html.
15. Patching policy for x64-based systems. http://www.microsoft.com/whdc/driver/kernel/64bitPatching.mspx.
16. Michaels Myers and Stephen Yound. An introduction to hardware-assisted virtual machine (hvm) rootkits, 2007. http://www.crucialsecurity.com/documents/hvmrootkits.pdf.
17. http://uninformed.org/index.cgi?v=3.
18. PNA. How to hook any api function in kernel32.dll, 1998. http://www.woodmann.com/yates/SystemHooking/export.htm.
19. Gerald J. Popek and Robert P. Goldberg. Formal requirements for virtualizable third generation architectures. In *Communications of the ACM,v.17 n.7, p.412-421*, July 1974.

20. J. Robin and C. Irvine. Analysis of the intel pentium's ability to support a secure virtual machine monitor. In *In Proceedings of the 9th USENIX Security Symposium, Denver, CO, August 2000.*, 2000.
21. Mark Russinovich. Sony, rootkits and digital rights management gone too far, 2005. `http://blogs.technet.com/markrussinovich/archive/2005/10/31/sony-rootkits-and-digital-rights-management-gone-too-far.aspx`.
22. J. Rutkowska and A. Tereshkin. Blue pill project. `http://bluepillproject.org`.
23. Joanna Rutkowska. Introducing stealth malware taxonomy. In *Black Hat*, 2006. `http://www.invisiblethings.org/papers/malware-taxonomy.pdf`.
24. Joanna Rutkowska. Isgameover(), anyone? In *Presentation on Black Hat Conference, Vegas*, 2007. `https://www.blackhat.com/presentations/bh-usa-07/Rutkowska/Presentation/bh-usa-07-rutkowska.pdf`.
25. Ken Thompson. Reflections on trusting trust. In *Commun. ACM,27(8) :761-763*, 1984. `http://cm.bell-labs.com/who/ken/trust.html`.
26. K. Tocheva, G. Erdelyi, A. Podrezov, S. Rautiainen, and M. Hypponen. Analysis of the nimda worm, September 2001. `http://www.f-secure.com/v-descs/nimda.shtml`.
27. Futo. `http://www.uninformed.org/?v=3\&a=7\&t=sumry`.
28. D. A. D. Zovi. Hardware virtualization rootkits. In *Black Hat USA*, 2006. `http://www.blackhat.com/presentations/bh-usa-06/BH-US-06-Zovi.pdf`.

Trusted Software within Focal

Philippe Ayrault[1], Matthieu Carlier[2], David Delahaye[3], Catherine Dubois[2], Damien Doligez[4], Lionel Habib[1], Thérèse Hardin[1], Mathieu Jaume[1], Charles Morisset[5], François Pessaux[1], Renaud Rioboo[2], and Pierre Weis[4]

[1] Université Pierre et Marie Curie-Paris 6, SPI-LIP6,
104 avenue du Président Kennedy, Paris 75016, France,
Firstname.Lastname@lip6.fr
[2] ENSIIE-CEDRIC
1 Square de la résistance, 91025 Evry Cedex, France,
Lastname@ensiie.fr
[3] CNAM-CEDRIC
292 rue Saint Martin, 75003, Paris, France
Firstname.Lastname@cnam.fr
[4] INRIA
Bat 8. Domaine de Voluceau, BP 105, F-78153 Le Chesnay, France,
Firstname.Lastname@inria.fr
[5] United Nations University, International Institute for Software Technologies,
UNU-IIST, P.O. Box 3058, Macao SAR,
Lastname@iist.unu.edu.fr

Abstract This paper describes the Integrated Development Environment Focal together with a brief proof of usability on the formal development of access control policies. Focal is an IDE providing powerful functional and object-oriented features that allow to formally express specification and to go step by step (in an incremental approach) to design and implement while proving that the implementation meets its specification or design requirements. These features are particularly well-suited to develop libraries for secure applications.

1 Introduction

Since at least forty years, an important area of software engineering is concerned with safety of industrial systems as those critical systems use more and more software components. Since twenty years, security problems of information systems spread from military domain to all society activities. Some fifteen years ago, safety and security of systems were usually considered of separated concern. Now, although their concerns are different, it is recognized that, in most cases, these two families of requirements must be considered together to receive satisfactory solutions (see the new rules for SCADA systems for example). Moreover, some methods to deal with safety requirements can be adapted to security requirements and conversely. Whatever is the domain, their methods are evolving, ad-hoc and empirical approaches being replaced by more formal methods. For example, for high levels of safety, formal models of the requirement/specification phase are more and more considered as they allow mechanized proofs, test or static analysis of the required properties. In the same way, high level assurance in system security asks for the use of true formal methods along the process of software development and is often required for the specification level.

To ease developing high integrity systems with numerous software components, an Integrated Development Environment (IDE) must provide tools to formally express specifications, to describe design and coding and to ensure that specification requirements are met by the corresponding code. This is not enough. First, standards of critical systems ask for pertinent documentation which has to be maintained along all the revisions during the system life cycle. Second, the evaluation conformance process of software is by nature a sceptical analysis. Thus, any proof of code correctness must be easily redone at request and traceability must be eased. Third, design and coding are difficult tasks. Research in software engineering has demonstrated the help provided by some object-oriented features as inheritance, late binding and early research works on programming languages have pointed out the importance of abstraction mechanism such as modularity to help invariant maintaining. There are a lot of other points which should also be considered when designing an IDE for safe and/or secure systems to ensure conformance with high Evaluation Assurance or Safety Integrity Levels (EAL-5,7 or SIL 3,4) and to ease the evaluation process according to various standards (e.g. IEC61508, CC, ...) : handling of non-functional contents of specification, handling of dysfunctional behaviors and vulnerabilities from the true beginning of development and fault avoidance, fault detection by validation testing, vulnerability and safety analysis.

The main aim of this paper is to present an IDE, called Focal [24,9] (freely distributed at `http://focal.inria.fr`), dedicated to the complete development of high integrity software, which attempts to give a positive solution to the three requirements identified above, the other items being currently under consideration. It provides means for the developers to formally express their specifications and to go step by step (in an incremental approach) to design and implementation while proving that such an implementation meets its specification or design requirements. It also provides some automation of documentation production and management.

Focal has already been used to develop huge examples. First, a computer algebra library was developed by Rioboo [27], it offers full specification and implementation of usual algebraic structures up to multivariate polynomial rings with complex algorithms. The point was to measure how Focal can help to render mathematical specifications and also to measure efficiency of the produced code, which is comparable (even little better) to the best general computer algebra systems in existence. Such a library is very useful when formalising the algebra of access control models, using implementations of orderings, lattices and boolean algebras (presented below). Focal was also very successfully used to formalise airport security regulations [7].

Focal semantics was initially specified in Coq, which brings a satisfactory confidence in the language's correctness. On the other side, the correction of the compiler against Focal's semantics is proved (by hand) [23].

In a second part, we will expose the usability of Focal through the formalization and development of access control policies. We present how formal methods can be used in practice to obtain trusted implementations of an access control policy. The library of access control policies offers a generic specification of the model and several developments based on it, giving implementations of classic access control policies.

2 The Focal Philosophy

Since our aim is to have a unique framework from specification to implementation, from code to proofs of requirements, Focal provides a unique and manifold language to express all theses aspects of software development.

Before really entering inside Focal, we briefly remind a few well-known concepts in term of software engineering.

Specifications of a system describe its functionalities, without referring to any particular practical solution. On the other side, the implementation gives an explicit, algorithmic, solution making the system running.

The Statement of Work of critical systems require the holding of certain properties (often called "requirements") to ensure that the system will be indeed operational. All along the development cycle, such requirements must be expressed and verified. Moreover, each development stage may introduce its own requirements. For instance, at specification-time, a railway system may require that doors of a train can't be opened while running. At implementation-time, this requirement must still hold, but some extra ones, due to implementation constraints may arise. For instance, the fact that a square root function is only called on positive numbers in the speed computation. This requirement was previously hidden since the speed comparison was not expressed finely enough to make the square root function appearing.

Considering these concepts, Focal allows management of declarations (specification), algorithms (implementation), properties (requirements) and proofs (demonstrations that requirements hold).

2.1 The Basic Brick

The primitive entity of a Focal development is the *species*. It can be viewed as a record grouping "things" related to a same concept. Like in most modular design systems (i.e. objected oriented, algebraic abstract types) the idea is to group a data structure with the operations to process it. Since in Focal we don't only address data type and operations, among these "things" we also find the specification of these operations, the representation of requirements (properties) and their proofs.

We now describe each of these "things", called *methods*.

- The *method* introduced by the keyword `rep` gives the data representation (*carrier*) that the *species* embeds. It is a type called the *carrier type* and defined by a type expression. The *carrier* may be not-yet-defined in a *species*, meaning that the real structure of the datatype the *species* embeds does not need to be known at this point. In this case, it is represented by a type variable. However, to obtain an implementation, the *carrier* has to be defined later either by setting `rep = exp` where `exp` is a type expression or by inheritance (see below). Type expressions in Focal are roughly ML-like types (variables, basic types, inductive types, record types) plus *species carrier types*, denoted by keyword `Self` inside the species and by the name of their *species* outside of them.
  Each *species* has a unique method *rep* and thus, a unique *carrier*. This is not a restriction compared to other object-oriented languages where an object can own several private variables representing the internal state, hence the data structure of the object. In such a case, the *carrier* type can simply be the tuple grouping all these variables that were disseminated all along the object.
- Declarations (`signature`) introducing a name and a type allows to announce a *method* to be defined later, i.e. to only specify its type, without implementation yet. Such *methods* are especially dedicated for specification or design purposes since it allows to use the name of the introduced method to define others *methods* while delaying the choice of its implementation. The type provided by the *signature* allows Focal to ensure via type-checking that the method is used in contexts compatibles with this type. The late-binding and the collection mechanisms, further introduced, ensure that the definition of the method will be effectively known when needed.

- Definitions (`let`) made of a name, a type and an expression introduce functions, i.e. computational operations. The core language used to implement them is roughly ML-like expressions (let-binding, pattern matching, conditional, higher order functions, ...) with the addition of a construction to call a *method* from a given *species*. Mutually recursive definitions are introduced by `let rec`.
- Statements (`property`) introduce a name and a first-order formula. A *property* may serve to express requirements (i.e. facts that the system must hold to conform to the Statement of Work delivered by the customer) and then can be viewed as a specification purpose *method*, like *signature*s were for `let`-*method*s. It will lead to a proof obligation later in the development. A *property* may also be used to express some "quality" information of the system (soundness, correctness, ..) also submitted to a proof obligation. Formulae are written with usual logical connectors, universal and existential quantifications over a Focal type, and names of *methods* known within the *species*'s context. For instance, a *property* telling that if a speed is non-null, then doors can't be opened could look like :
  `all v in Speed, v <> Speed!zero -> ~ doors_open`
  In the same way as *signatures*, even if no proof is yet given, the name of the *property* can be used to express other ones and its statement can be used as an hypothesis in proofs. Focal late binding and collection mechanisms ensure that the proof of a *property* will be ultimately done.
- Theorems (`theorem`) made of a name, a statement and a proof are *properties* together with the formal proof that their statement holds in the context of the *species*. The proof accompanying the statement will be processed by Focal and ultimately checked with the theorem prover Coq.
  Like in any formal development, one severe difficulty before proving is obviously to enounce a true interesting and meaningful statement. For instance, claiming that a piece of software is "formally proved" as respecting the safety requirements `system_ok` "since **its** property is demonstrated" is a lie if this property was, for instance, `1 = 1 -> system_ok`. This is obviously a non-sense since the text of the property is trivial and does not link `system_ok` with the rest of the software (see [10] for less trivial examples).

We now make concrete these notions on an example we will incrementally extend. We want to model some simple algebraic structures. Let's start with the description of a "setoid" representing the data structure of "things" belonging to a set, which can be submitted to an equality test and exhibited (i.e. one can get a witness of existence of one of these "things").

```
species Setoid =
  signature ( = ) : Self -> Self -> bool ;
  signature element : Self ;

  property refl : all x in Self, x = x ;
  property symm : all x y in Self, x = y -> y = x ;
  property trans: all x y z in Self, x=y and y=z -> x=z ;
  let different (x, y) = basics#not_b (x = y) ;

end ;;
```

In this *species*, the *carrier* is not explicitely given (no `rep`) , since we don't need to set it to be able to express functions and properties our "setoid" requires. However, we can refer to it via `Self` and it is in fact a type variable. In the same way, we specify a *signature* for the equality (operator `=`). We introduce the three properties that an equality (equivalence relation) must conform to.

We complete the example by the definition of the function `different` which use the name `=` (here `basics#not_b` stands for the function `not_b`, the boolean `and` coming from the Focal source file `basics.foc`). It is possible right now to prove that `different` is irreflexive, under

the hypothesis that = is an equivalence relation (i.e. that each implementation of = given further will satisfy these properties).

It is possible to use *methods* only declared before they get a real *definition* thanks to the *late-binding* feature provided by Focal. In the same idea, redefining a *method* is allowed in Focal and, it is always the last version which is kept as the effective *definition* inside the species.

2.2 Type of Species, Interfaces and Collections

The type of a *species* is obtained by removing definitions and proofs. If `rep` is still a type variable say α, then the *species* type is prefixed with an existential binder $\exists\alpha$. This binder will be eliminated as soon as the `rep` will be instantiated (defined) and must be eliminated to obtain runnable code. The species types remain implicit in the concrete syntax.

The *interface* of a species is obtained by abstracting the `rep` type in all the method types of the species type and this abstraction is permanent (see the paragraph Collections). No special construction is given to denote *interfaces* in the concrete syntax, they are simply denoted by the name of the species underlying them. Interfaces can be ordered by inclusion, a point providing a very simple notion of subtyping.

A species is said to be *complete* if all declarations have received definitions and all properties have received proofs.

When *complete*, a species can be submitted to an abstraction process of its carrier to create a *collection*. Thus the interface of the collection is built out of the type of its underlying species. A collection can hence be seen as an abstract data type, only usable through the methods of its interface, but having the guarantee that all methods/theorems are defined/proved.

2.3 Combining Bricks by Inheritance

A Focal development is organised as a hierarchy which may have several roots. The upper levels of the hierarchy are built during the specification stage while the lower ones correspond to implementations. Each node of the hierarchy, i.e. each *species*, is a progress to a complete implementation. On the previous example, forgetting `different`, we typically presented a kind of *species* for "specification" since it expressed only *signatures* of functions to be later implemented and properties to which, later, give *proofs*.

We can now create a new *species*, may be more complex, by **inheritance** of a previously defined. We say here "may be more complex" because it can add new operations and properties, but it can also only bring real definitions to *signatures* and *proofs* to *properties*, adding no new *method*.

Hence, in Focal inheritance serves two kinds of evolutions. In the first case the evolution aims making a *species* with more operations but keeping those of its parents (or redefining some of them). In the second case, the *species* only tends to be closer to a "run-able" implementation, providing explicit definitions to *methods* that were previously only declared. A strong constraint in inheritance is that the type of inherited, and/or redefined *methods* must not change. This is required to ensure consistence of the Focal model, hence of the developed software.

Continuing our example, we want to extend our model to represent "things" with a multiplication and a neutral element for this operation.

```
species Monoid inherits Setoid =
  signature ( * ) : Self -> Self -> Self ;
  signature one : Self ;
  let element = one * one ;
end ;;
```

We see here that we added new *methods* but also gave a definition to `element`, saying it is the application of the method `*` to `one` twice, both of them being only *declared*. Here, we used the inheritance in both the presented ways : making a more complex entity by adding *methods* and getting closer to the implementation by explicitly defining `element`.

Multiple inheritance is available in Focal. For sake of simplicity, the above example uses simple inheritance. In case of inheriting a *method* from several parents, the order of parents in the `inherits` clause serves to determine the chosen *method*.

The type of a *species* built using inheritance is defined like for other *species*, the *methods* types retained inside it being those of the *methods* present in the *species* after inheritance is resolved.

2.4 Combining Bricks by Parameterisation

Until now we are only able to enrich *species* (may be "refine", even if we do not address the notion of "refinement" of the Atelier B [1]). However, we sometimes need to use a *species*, not to take over its *methods*, but rather to use it as an "ingredient" to build a new structure. For instance, a pair of setoids is a new structure, using the previous *species* as the "ingredient" to create the structure of the pair. Indeed, the structure of a pair is independent of the structure of each component it is made of. A pair can be seen as *parameterised* by its two components. Following this idea, Focal allows two flavors of parameterisation.

Parameterisation by Collection Parameters We first introduce the *collection parameters*. They are *collections* that the hosting species may use through their *methods* to define its own ones.

A *collection parameter* is given a name C and an interface I. The name C serves to call the *methods* of C which figure in I. C can be instantiated by an effective parameter CE of interface IE. CE is a collection and its interface IE must contain I. Moreover, the collection and late-binding mechanisms ensure that all methods appearing in I are indeed implemented (defined for functions, proved for properties) in CC. Thus, no runtime error, due to linkage of libraries, can occur and any *property* stated in I can be safely used as an hypothesis.

Calling a *species*'s *method* is done via the "bang" notation `:!meth` or `Self!meth` for a *method* of the current *species* (and in this case, even simpler : `meth`, since the Focal compiler will resolve scoping issues). To call *collection parameters*'s *method*, the same notation is used : `A!element` stands for the *method* `element` of the *collection parameter* `A`.

To go on with our example, a pair of setoids has two components, hence a *species* for pairs of setoids will have two *collection parameters*. It is itself a setoid, a fact which is simply recorded via the inheritance mechanism : `inherits Setoid` gives to `Setoid_product` all the methods of `Setoid`.

```
species Setoid_product (A is Setoid, B is Setoid) inherits Setoid =
  rep = (A * B) ;

  let ( = ) (x, y) =
     and_b
```

```
        (A!( = ) (first (x), first (y)),
         B!( = ) (scnd (x), scnd (y))) ;
  let create (x, y) in Self = basics#crp (x, y) ;
  let element = Self!create (A!element, B!element) ;

  proof of refl = by definition of ( = ) ;
end ;;
```

We express the *carrier* of the product of two setoids as the Cartesian product of the *carriers* of the two parameters. In `A * B`, `*` is the Focal type constructor of pairs, `A` denotes indeed the carrier type of the first *collection parameter*, and `B` the one of of the second *collection parameter*.
Next, we add a definition for `=` of `Setoid_product`, relying on the methods `=` of `A` (`A!( = )`) and `B` (which are not yet defined). Similarly, we introduce a definition for `element` by building a pair, using the function `create` (which calls the predefined function `basics#crp`) and the methods `element` of respectively `A` and `B`. And we can prove that `=` of `Setoid_product` is indeed reflexive, upon the hypothesis made on `A!( = )` and `B!( = )`. The part of Focal used to write proofs will be shortly presented later, in section 2.6.
This way, the *species* `Setoid_product` builds its *methods* relying on those of its *collection parameters*. Note the two different uses of `Setoid` in our *species* `Setoid_product`, which inherits of `Setoid` and is parameterised by `Setoid`.

Why such *collection parameters* and not simply *species parameters*? There are two reasons. First, effective parameters must provide definitions/proofs for all the methods of the required interface : this is the contract. Thus, effective parameters must be *complete* species. Then, we do not want the parameterisation to introduce dependencies on the parameters' *carrier* definitions. For example, it is impossible to express " if `A!rep` is `int` and `B!rep` is `bool` then `A*B` is a list of boolean values". This would dramatically restrict possibilities to instantiate parameters since assumptions on the *carrier*'s structure, possibly used in the parameterised *species* to write its own *methods*, could prevent *collections* having the right set of *methods* but a different internal representation of the *carrier* to be used as effective parameters. Such a behaviour would make parameterisation too weak to be usable. We choose to always hide the *carrier* of a *collection parameter* to the parameterised hosting *species*. Hence the introduction of the notion of *collection*, obtained by abstracting the carrier from a complete species.

Parameterisation by Entity Parameters Let us imagine we want to make a *species* working on natural numbers modulo a certain value. In 5 `modulo` 2 *is* 1, both 5 and 2 are natural numbers. To be sure that the *species* will consistently work with the same modulo, this last one must be embedded in the *species*. However, the *species* itself doesn't rely on a particular value of the modulo. Hence this value is clearly a **parameter** of the species, but a parameter in which we are interested by its **value**, not only by its *carrier* and the methods acting on it. We call such parameters *entity parameters*, their introduction rests upon the introduction of a *collection parameter* and they denote a *value* having the type of the *carrier* of this *collection parameter*.
Let us first have a *species* representing natural numbers :

```
species IntModel =
  signature one : Self ;
  signature modulo : Self -> Self -> Self ;
end ;;
```

Note that `IntModel` can be later implemented in various ways, using Peano's integers, machine integers, arbitrary-precision arithmetic ...
We now build our *species* "working modulo ... ", embedding the value of this modulo like :

```
species Modulo_work (Naturals is IntModel, n in Naturals) =
  let job1 (x in Naturals) in ... =
    ... Naturals!modulo (x, n) ... ;
  let job2 (x in Naturals, ...) in ... =
    ... ... Naturals!modulo (x, n) ... ... ;
end ;;
```

Using the *entity parameter* `n`, we ensure that the *species* `Modulo_work` will work for *any* value of the modulo, but will always use the *same* value `n` of the modulo everywhere inside the *species*.

2.5 The Final Brick

As briefly introduced in 2.2, a *species* needs to be fully defined to lead to executable code for its functions and checkable proofs for its theorems. When a *species* is fully defined, it can be turned into a *collection*, that can roughly be seen as an instance of the *species*. Hence, a *collection* represents the final stage of the inheritance tree of a *species* and leads to an effective structure of the *carrier* with executable functions processing it.

For instance, providing that the previous *species* `IntModel` turned into a fully-defined species `MachineNativeInt` through inheritances steps, with a *method* `from_string` allowing to create the natural representation of a string, we could get a related collection by :

```
collection MachineNativeIntColl implements MachineNativeInt ;;
```

Next, to get a *collection* implementing arithmetic modulo 8, we could extract from the *species* `Modulo_work` the following *collection* :

```
collection Modulo_8_work implements Modulo_work
   (MachineNativeIntColl, MachineNativeIntColl!from_string ("8") ;;
```

As seen by this example, a species can be applied to effective parameters by giving their values with the usual syntax of parameter passing.

As said before, to ensure modularity and abstraction, the *carrier* of a *collection* turns hidden. This means that any software component dealing with a *collection* will only be able to manipulate it through the operations (*methods*) it provides. This point is especially important since it prevents other software components from possibly breaking invariants required by the internals of the *collection*.

2.6 *Properties*, *Theorems* and *Proofs*

Focal aims not only to write programs, it intends to encompass both the executable model (i.e. program) and properties this model must satisfy. For this reason, "special" *methods* deal with logic instead of purely behavioral aspects of the system : *theorems*, *properties* and *proofs*.

Stating a *property* expects that a *proof* that it **holds** will finally be given. For *theorems*, the *proof* is directly embedded in the *theorem*. Such proofs must be done by the developer and will finally be sent to the formal proof assistant Coq who will automatically check that the demonstration of the *property* is consistent. Writing a proof can be done in several ways.

It can be written in "Focal's proof language", a hierarchical proof language that allows to give hints and directions for a proof. This language will be sent to an external theorem prover, Zenon [5,8] developed by D. Doligez. This prover is a first order theorem prover based on the tableau method incorporating implementation novelties such as sharing. Zenon will attempt,

from these hints to automatically generate the proof and exhibit a Coq term suitable for verification by Coq. Basic hints given by the developer to Zenon are : "prove by definition of a *method*" (i.e. looking inside its body) and "prove by *property*" (i.e. using the logical body of a *theorem* or *property*". Surrounding this hints mechanism, the language allows to build the proof by stating assumptions (that must obviously be demonstrated next) that can be used to prove lemmas or parts for the whole property.
The detailed presentation of Zenon, its internals and its language (the one we called "Focal's proof language") is outside the scope of this presentation. We however show below an example of such demonstration.

```
theorem order_inf_is_infimum: all x y i in Self,
  !order_inf(i, x) -> !order_inf(i, y) ->
    !order_inf(i, !inf(x, y))
  proof:
    <1>1 assume x in Self, assume y in Self,
         assume i in Self, assume H1: !order_inf(i, x),
         assume H2: !order_inf(i, y),
         prove !order_inf(i, !inf(x, y))
      <2>1 prove !equal(i, !inf(!inf(i, x), y))
        by hypothesis H1, H2
           property inf_left_substitution_rule,
             equal_symmetric, equal_transitive
           definition of order_inf
      <2>9 qed
        by step <2>1
           property inf_is_associative, equal_transitive
           definition of order_inf
    <1>2 qed.
  ;
```

The important point is that Zenon works for the developer : **it searches the proof itself**, the developer does not have to elaborate it formally "from scratch".
Like any automatic theorem prover, Zenon may fail finding a demonstration. In this case, Focal allows to write verbatim Coq proofs. In this case, the proof is not anymore automated, but this leaves the full power of expression of Coq to the developer.
Finally, the `assumed` keyword is the ultimate proof backdoor, telling that the proof is not given but that the property must be admitted. Obviously, a really safe development should not make usage of such "proofs" since they bypass the formal verification of software's model. However, such a functionality remains needed since some of "well-known" properties can never be proved for a computer. For instance, $\forall x \in \mathbb{N}, x + 1 > n$ does not hold in a computer with native integers : in this case arithmetic works modulo the number of bits of the machine word ! However, in a mathematical framework, this property holds and is needed to carry out other proofs ! On another side, a development may be linked with external code, trusted or not, but for which properties can't be proved inside the Focal part since it does not belong to it. Expressing properties of the Focal part may need to express properties on the imported code, that cannot be formally proved, then must be "assumed".

2.7 Around the Language

In the previous sections, we presented Focal through its programming model and shortly its syntax. We especially investigated the various entities making a Focal program. We now address what becomes a Focal program once compiled. We recall that Focal supports the redefinition of functions, which permits for example to specialize code to a specific representation of the carrier (for example, there exists a generic implementation of integer addition modulo `n` but it can be redefined in arithmetics modulo 2 if boolean values are used to represent the two values). It is also a very convenient tool to maintain software.

Consistency of the Software All along the development cycle of a Focal program, the compiler keeps trace of dependencies between *species*, their *methods*, the *proofs*, ... to ensure that modifications of an entity will be detected on any of those depending of the former.

Focal considers two types of dependencies :

- The **decl**-dependency : a *method* A decl-depends on a *method* B, if the **declaration** of B is required to write A.
- The **def**-dependency : a *method* (and more especially, a *theorem*) A def-depends on a *method* B, if the **definition** of B is required to write A (and more especially, to prove the property stated by the *theorem* A).

The redefinition of a function may invalidate the proofs that use properties of the body of the redefined function. All the proofs which truly depend of the definition are then erased by the compiler and must be done again in the context updated with the new definition. Thus the main difficulty is to choose the best level in the hierarchy to do a proof. In [25], Prevosto and Jaume propose a *coding style* to minimize the number of proofs to be redone in the case of a redefinition, by a certain kind of modularisation of the proofs.

Code Generation Focal currently compiles programs toward two languages, OCaml to get an executable piece of software, and Coq to have a formal model of the program, with theorems and proofs.

In OCaml code generation, all the logical aspects are discarded since they do not lead to executable code.

Conversely, in Coq, all the *methods* are compiled, i.e. "computational" *methods* and logical *methods* with their proofs. This allows Coq to check the entire consistence of the system developed in Focal.

Since Focal's compilation model is based on record types (i.e. `struct` *à la* C) code generation toward most of the current programming languages can be imagined. A C backend is currently under study, which should allow to manage once for all the compilation of high-level constructs found in ML-like languages and reused in Focal (pattern-matching, higher-order functions, variant types ...).

Tests Focal incorporates a tool named FocalTest [20] for Integration/Validation testing. It allows to confront automatically a property of the specification with an implementation. It generates automatically test cases, executes them and produces a test report as an XML document. The property under test is used to generate the test cases, it also serves as an oracle. When a test case fails, it means a counterexample of the property has been found : the implantation does not match the property ; it can also indicate an error in the specification.

The tool FocalTest automatically produces the test environment and the drivers to conduct the tests. We benefit from the inheritance mechanism to isolate the testing harness from the components written by the programmer.

The testable properties are required to be broken down into a precondition and a conclusion, both executable. FocalTest proposes a pure random test cases generation : it generates test cases until the precondition is satisfied, the verdict of the test case is given by executing the postcondition. It can be an expensive process for some kind of preconditions. To overcome this drawback, a constraint based generation is under development : it allows to produce directly test cases for which the precondition is satisfied.

Documentation The tool called FOCDOC [19] automatically generates documentation, thus the documentation of a component is always coherent with respect to its implementation.

This tool uses its own XML format that contains information coming not only from structured comments (that are parsed and kept in the program's abstract syntax tree) and Focal concrete syntax but also from type inference and dependence analysis. From this XML representation and thanks to some XSLT stylesheets, it is possible to generate HTML files or LaTeX files. Although this documentation is not the complete safety case, it can helpfully contribute to its elaboration. In the same way, it is possible to produce UML models [6] as means to provide a graphical documentation for Focal specifications. The use of graphical notations appears quite useful when interacting with end-users, as these tend to be more intuitive and are easier to grasp than their formal (or textual) counterparts. This transformation is based on a formal schema and captures every aspect of the Focal language, so that it has been possible to prove the soundness of this transformation (semantic preservation).
Focal's architecture is designed to easily plug third-parties analyses that can use the internal structures elaborated by the compiler from the source code. This allows, for example, to make dedicated documentation tools for custom purposes, just exploiting information stored in the Focal program's abstract syntax tree, or extra information possibly added by extra processes, analyses.

3 Access control : a Case Study within Focal

Access control is any mechanism by which a system grants or revokes the rights for active entities, the subjects, to access some passive entities, the objects, or perform some action. In this section we present a brief survey of the library of access control models developed within Focal.
Several approaches have been followed in order to build a formal library of access control models. First, in [11], we have formalised within Coq [26] the Bell & LaPadula (BLP) model by following the original paper [18] : definition of the policy, of the transition function between states of the system and formal proof of the "Basic Security Theorem" [18] stating that the transition function preserves the policy. Then, by using the program extraction (from a proof) mechanism of Coq, we have obtained a certified implementation of this access control policy. Such development formally ensures that the program we have obtained satisfies the desired security properties, thus providing a greater level of confidence. However, it is rather technical and time-consuming and requires some backgrounds within Coq. Furthermore, this formalisation cannot be easily reused if we want to implement another policy. Indeed many policies share some definitions and properties and, as it is widely recognised, it seems desirable to deal with an abstract generic framework in order to ease and speed implementations by reusing. Indeed, having an abstract formalism would allow to obtain an implementation which is well-suited for a given context just by instantiating parameters. Hence, in a second time, we have used Focal to implement the algebra of security model introduced by J.McLean in [21]. Such algebra provides a generic framework to specify policies. From this implementation, we have instantiated the framework in order to obtain a formal development of the BLP model [13,14]. In order to illustrate the practical applicability of the previous development, we have refined this system to manage accesses into a relational database. To achieve this goal, we do not want to define the complete database in Focal, but rather to define a reference monitor and to plug it into an existing database, such as MySQL. So we have to catch every SQL query sent to the server and translate it in terms of access, thus providing requests for the access control system. Then these requests are executed by our Focal program, which returns an answer. If this answer is yes, then the SQL query is executed by the SQL server, else an error message is returned to the user (see figure 1). Of course, more sophisticated security models exist for databases, but our aim was to show that our "formal program" can be used in concrete contexts. All this work is described in [3,4].

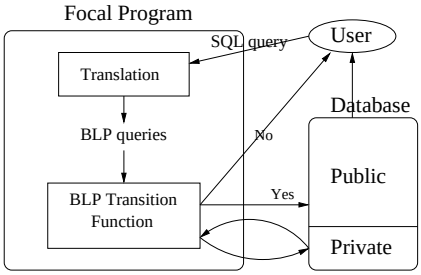


FIG. 1: Access control into a database

Unfortunately, the framework of the "algebra of security" is not enough expressive for some access control models and we have defined and implemented within Focal a specification of access control policies at a deeper level. Hence, we have addressed the question of "what is an access control model?" regardless of any specific context, by defining a general semantical framework allowing to specify and to define access control models. Such a framework provides several levels of specification. First, the considered information system can be described by specifying the security parameters and by defining how to represent states of this system (what is the security information describing a state of the system). Then an access control policy is defined as a means to permit or deny a subject (users, processes, ...) the use of an object (files, processes, ...). This can be done by introducing a predicate allowing to characterise secure states of the system (which are the states satisfying the policy). The next step consists in specifying the syntax and the semantics of a language of requests allowing the system to move from one state to another state. The specification of a policy and a language of requests leads to the notion of access control model. Lastly, our framework allows to describe what are the transition functions (definitions of reference monitors) of a model and what security properties these transition functions have to satisfy (the main properties are concerned with the policy and the semantics of requests). Furthermore, since several transition functions can be defined for an access control model, we introduce a way to compare such transition functions.

As a usage of Focal, we are developing an access control library based on the framework previously described. The architecture of the framework's implementation within Focal is shown in figure 2. As we consider the systems on which we apply access control as state transition systems (reference monitors are transition functions), we see that the species `states` is the central species of the implementation. Indeed, it is parameterised by the basic components of an access control system (subjects, objects, security parameter, ...), and is inherited by the species defining the access control policy (`policies`), the semantics of requests (`semantics_requests`) and the transition function (`models`). We also see how we use the framework's implementation in order to implement a particular access control model, this is done by refining the framework's species thanks to inheritance. Those species defining an access control model are the ones ending with a `*`, where `*` can be replaced by the name of the access control model implemented. For instance, the library currently offers the Bell & LaPadula, RBAC, HRU, Unix-like and Unforgeable Tickets (capabilities based) models. Hence, this architecture shows some of the strength of Focal which are parameterisation and refinement through inheritance that allow us to follow a quite complex formalisation and to implement access control models by first considering what is common to every models and

then focusing on what is particular to a specific one (this way of developing also ease code reusing).

Another major feature of Focal is to allow to write *properties* and to prove them in order to show the correctness of an implementation. This has currently be done in the access control library for the RBAC and HRU models (our aim is to certify all the implemented models), and we show here an example of such a theorem concerning the correctness of the RBAC transition function. Indeed, the following theorem `tau_rbac_acc_secure` states the correctness of the RBAC transition function according to the RBAC security predicate (`omega`). For the sake of simplicity, we only consider here the requests allowing to ask to get or to release an access (the correctness proof taking into account the administrative requests of the RBAC model has also be done).

```
theorem tau_rbac_acc_secure :
  all st1 st2 in Self, all r1 in Req, all d1 in Dec,
    tau_rbac_acc(r1, st1) = (d1, st2) ->
      omega(st1) ->
        omega(st2)
```

We now show the architecture of its proof. The first step consists to consider the hypothesis needed to do the proof and to express what we want to prove, as we would do in any mathematical proof.

```
  proof:
    <1>1 assume st1 in Self,
         assume st2 in Self,
         assume r1 in Req,
         assume d1 in Dec,
         assume H1: tau_rbac_acc(r1, st1) = (d1, st2),
         assume H2: omega(st1),
         prove omega(st2)
```

Then, mainly, we need to consider two cases corresponding to the two possible requests, "get an access" (`is_get`) or "release an access" (`is_rel`) and prove our goal. Finally, we are able to conclude this proof thanks to the property `get_or_rel` stating that the requests we consider are either "get an access" or "release an access" requests.

```
      <2>2 assume H3: Req!is_get(r1),
           prove omega(st2)
        ...
      <2>3 assume H4: Req!is_rel(r1)
           prove omega(st2)
        ...
      <2>f qed by step <2>2, <2>3
                  property Req!get_or_rel
    <1>f qed. ;
```

To summerize, our framework, described in [15,16,17,22], has been used to formalise, to implement within Focal and to compare the BLP model [22], the Chinese Wall model [22], the RBAC (role based access control) model [12], and discretionary policies (HRU model, trust management model, unforgeable tickets model) [2].

4 Conclusion

This paper is mostly devoted to Focal which was conceived from the beginning to help constructions of systems highly concerned with safety and security. Its development is based on strong theories such as type theories, denotational and operational semantics, rewriting.

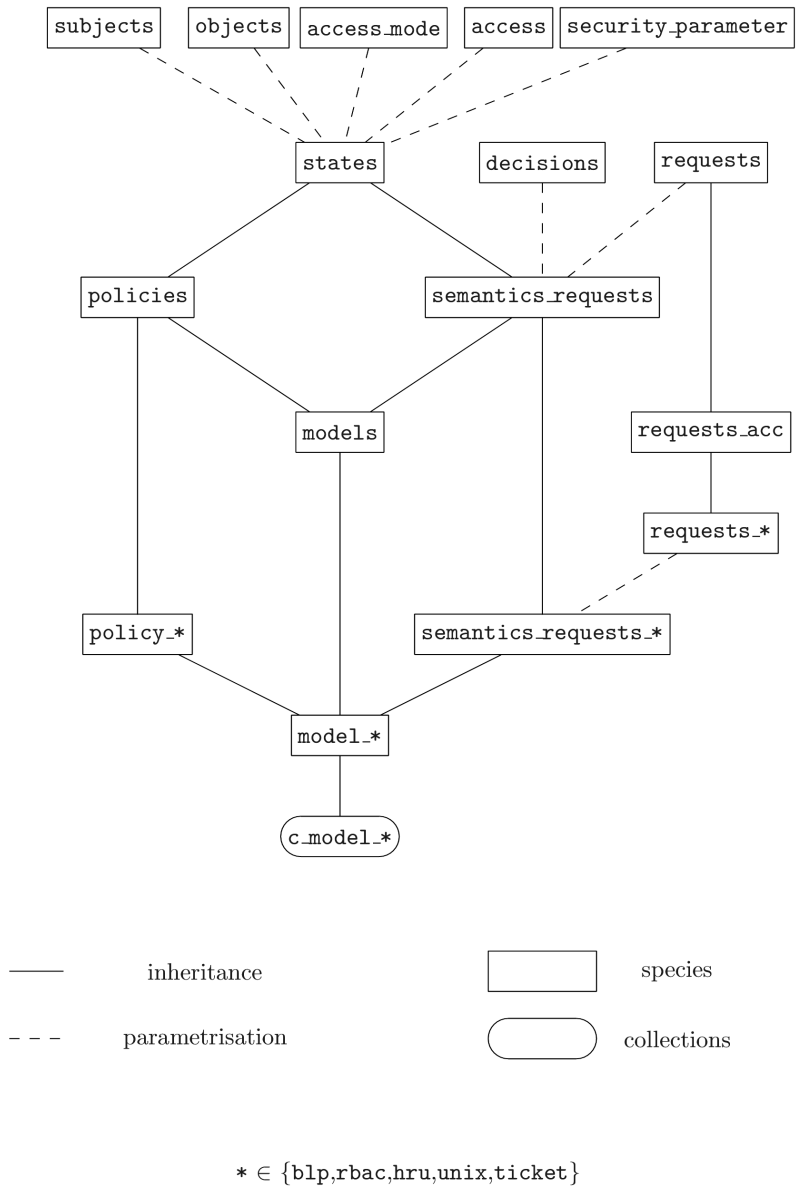


FIG. 2: Architecture of the framework's and access control models' implementation

But, our methodology for Focal development is to incorporate only parts of these theories which are mandatory for our purposes. For example, the Focal language is indeed a dependent type language but some dependencies available for example in the proof assistant Coq are not offered : for instance, a function cannot depend of a proof. If such a possibility is retained, then treatment of partiality of functions can perhaps be improved but we would have to manage possible logical clashes when redefining a function. It seems to us that this management would be too difficult for engineers and we reject this possibility.

The claim behind Focal is that formal developments tend to increase the confidence in the final code. One of the main characteristics of critical software is that it is subject to the approval of a safety/security authority before its commissioning. These authorities have defined requirements explaining what should be an acceptable software and its related life cycle process for their own domain. For this reason, getting a high confidence in produced code, and making possible for the safety/security authority to acquire this confidence is an important task, for which Focal brings solutions. Very important is the possibility in Focal to have one unique language for specification, implementation and proofs, since it eliminates the errors introduced between each layer, each switch between languages, during the development cycle.

Other frameworks like Atelier B [1] also aims to implement tools for making formal development a reality. Focal doesn't follow the same path, trying to keep the mean of expression close to what engineers usually know : something close to a programming language. Moreover, instead of having its own system for proofs validation, Focal makes use of external tools, leaving the task of handling proof automation and verification outside its scope and keeping benefits from researches performed aside in these specific domains.

Acknowledgments This work was partially supported by the French SSURF ANR project ANR-06-SETI-016 (Safety and Security UndeR Focal). Many thanks to James Geater for his careful proofreading.

Références

1. J. R. Abrial. *The* B *Book, Assigning Programs to Meanings.* Cambridge University Press, Cambridge (UK), 1996. ISBN 0521496195.
2. F. Anseaume, J. Baron, P. Berthelin, M. Jacquel, and D. Pipon. Formalisation, spécification et implantation de politiques de contrôle d'accès avec l'atelier focal. Master's thesis, UPMC, Paris, France, 2008.
3. J. Blond and C. Morisset. Formalisation et implantation d'une politique de sécurité d'une base de données. In INRIA, editor, *JFLA'2006*, pages 71–86, 2006.
4. J. Blond and C. Morisset. Un moniteur de référence sûr d'une base de données. *Technique et Science Informatiques*, 26(9) :1091–1110, 2007.
5. Richard Bonichon, David Delahaye, and Damien Doligez. Zenon : An Extensible Automated Theorem Prover Producing Checkable Proofs. In *Logic for Programming Artificial Intelligence and Reasoning (LPAR)*, volume 4790 of *LNCS/LNAI*, pages 151–165, Yerevan (Armenia), October 2007. Springer.
6. David Delahaye, Jean-Frédéric Étienne, and Véronique Viguié Donzeau-Gouge. A Formal and Sound Transformation from Focal to UML : An Application to Airport Security Regulations. In *UML and Formal Methods (UML&FM)*, Innovations in Systems and Software Engineering (ISSE) NASA Journal, Kitakyushu-City (Japan), October 2008. Springer.

7. David Delahaye, Jean-Frédéric Étienne, and Véronique Viguié Donzeau-Gouge. Formal Modeling of Airport Security Regulations using the Focal Environment. In *Requirements Engineering and Law (RELAW)*, Barcelona (Spain), September 2008. IEEE CS Press.
8. D. Doligez. Zenon, version 0.4.1. `http://focal.inria.fr/zenon/`, 2006.
9. Catherine Dubois, Thérèse Hardin, and Véronique Viguié Donzeau-Gouge. Building certified components within focal. In *Symposium on Trends in Functional Programming*, 2004.
10. E.Jaeger and T.Hardin. A few remarks about developing secure systems in b, 2008. Unpublished.
11. E. Gureghian, Th. Hardin, and M. Jaume. A full formalisation of the Bell and Lapadula security model. Technical Report 2003-007, Univ. Paris 6, LIP6, 2003.
12. L. Habib, M. Jaume, and C. Morisset. A formal comparison of the bell & lapadula and rbac models. In *Information Assurance and Security (IAS'08) International Conference on Information Technology, ITCC*, page To appear. IEEE CS Press, 2008.
13. M. Jaume and C. Morisset. Formalisation and implementation of access control models. In *Information Assurance and Security (IAS'05) International Conference on Information Technology, ITCC*, pages 703–708. IEEE CS Press, 2005.
14. M. Jaume and C. Morisset. A formal approach to implement access control. *Journal of Information Assurance and Security*, 2 :137–148, 2006.
15. M. Jaume and C. Morisset. Towards a formal specification of access control. In *Joint Workshop on Foundations of Computer Security and Automated Reasoning for Security Protocol Analysis FCS-ARSPA'06 (Satellite Workshop to LICS'2006)*, 2006.
16. M. Jaume and C. Morisset. Contrôler le contrôle d'accès : Approches formelles. In *Approches Formelles dans l'Assistance au Développement de Logiciels, AFADL'07*, 2007.
17. M. Jaume and C. Morisset. On specifying, implementing and comparing access control models. A Semantical Framework. Technical Report 2007, Univ. Paris 6, LIP6, 2007.
18. L.J. LaPadula and D.E. Bell. Secure Computer Systems : A Mathematical Model. *Journal of Computer Security*, 4 :239–263, 1996.
19. M. Maarek and V. Prevosto. Focdoc : The documentation system of foc. In *Proceedings of the 11th Calculemus Symposium*, Rome, sep 2003.
20. M.Carlier and C.Dubois. Functional testing in the focal environment. In B.Beckert and R.Hähnle, editors, *Tests and Proofs, Second International Conference, TAP 2008, Prato, Italy, April 9-11, 2008. Proceedings*, volume 4966 of *Lecture Notes in Computer Science*, pages 84–98. Springer, 2008.
21. McLean. The algebra of security. In *Proc. IEEE Symposium on Security and Privacy*, pages 2–7. IEEE Computer Society Press, 1988.
22. C. Morisset. *Sémantique des systèmes de contrôle d'accès*. PhD thesis, Université Pierre et Marie Curie - Paris 6, 2007.
23. V. Prevosto. *Conception et Implantation du langage FoC pour le développement de logiciels certifiés*. PhD thesis, Université Paris 6, sep 2003.
24. V. Prevosto and D. Doligez. Algorithms and proof inheritance in the Foc language. *Journal of Automated Reasoning*, 29(3-4) :337–363, dec 2002.
25. V. Prevosto and M. Jaume. Making proofs in a hierarchy of mathematical structures. In *Proceedings of the 11th Calculemus Symposium*, Rome, sep 2003.
26. Logical Project. *The Coq Proof Assistant Reference Manual*. INRIA-Rocquencourt, 2006.
27. R. Rioboo. *Programmer le Calcul Formel, des Algorithmes à la Sémantique*. PhD thesis, Univ. Paris 6, 2002.

Panorama of Secure Contactless Communications

François Vacherand, Elisabeth Crochon, François Dehmas, Jacques Reverdy, and Olivier Savry

CEA-LETI MINATEC, 17 rue des Martyrs F-38054 Grenoble Cedex 9
françois.vacherand@cea.fr

Abstract This paper introduces security issues of contactless technology. Basically, this technology, contrary to the traditional wireless technology for radio communication systems, is strongly dissymmetrical. Electromagnetic field and power are only emitted by readers towards passives transponders. Cards or tags transmit data through retro-modulation or back-scattering. Moreover, there is no ON/OFF switch on these passive devices and they can be activated remotely without any action of their owners. Transactions can thus start without the awareness of the users, which introduces a strong potential threat depending of the type of application. Finally, there is an over the air open channel, which is a new issue for smart cards. Some protections and countermeasures have to be introduced, at least for careful secure design rules. The application fields are mainly those of contactless smart card, RFID tags and NFC devices which belong to the trusted computing area but with the drastic characteristic of low resources devices. Keywords : Contactless, RFID, NFC, Secure Communications, Hacking, Privacy, Low Resources Devices, Trusted Computing, Risk Analysis

Introduction

General Overview

In the last decade, there was a tremendous growth of contactless applications. Teleticketing and more generally access control were the first large scale and public initiators, and then credit cards and electronic purse moved to this technology. Finally e-Identity such as National Identity card and e-Passport started national or worldwide applications. This emerging technology brought new and fair daily usage, but offered new ways of fraudulent or malicious actions directly related to the economical or security aspects of most of the applications. This paper focuses on the security of the contactless channel, mainly at the physical level, and on physical attacks and associated countermeasures. The contactless technology has particular characteristics which were not yet addressed in wireless radio technology. These two technologies share in common the data transmission through the RF channel, basically bidirectional transmissions. So far, the threats in term of security which jeopardize the transmission of information are a priori the same ones : remote listening or eavesdropping. However, there is an important difference : the contactless system has an active element, the reader, and a passive element, the card or the tag, which are remotely powered thanks to the electromagnetic field of the reader. This system, contrary to the traditional radiofrequency emitter/receiver couple, is thus strongly dissymmetrical. This dissymmetry is pointed out by the evidence that there is no ON/OFF switch on these passive devices and that they can be activated automatically without the help of their owner. Transactions can thus start without the awareness of

the users, which introduces a more or less strong potential threat depending of the type of application. Comparing to contact smart cards, it is worth to point out that the RF channel opens a new potential weakness by enabling and performing information communication over the air.

Paper organization

First, the article attempts to introduce the contactless technology and the secure aspects. Then a state of the art of the attacks relating to this technology is presented. It will focus as well on the scenarios related to the economical fraud and hacking as the scenarios on privacy issues. Finally some examples of countermeasures will be introduced. The application fields are mainly those of the contactless smart card, of the RFID tags and of the NFC devices.

Contactless systems

Introduction to contactless systems

The contactless systems addressed in this paper are based on communication of radiofrequency type and consist of a reader emitting an electromagnetic field and a passive contactless object collecting its energy from the electromagnetic field. This latter is also used to transfer data in both directions. These contactless systems are characterized by a trade-off between range and complexity. Basically the field emitted by the reader decreases with distance, thus the further away the portable object, the lower the power supply and the smaller the power of computation of the embedded processing.

For example, RFID identification tags will give priority to operating distance : from a few centimeters to a few meters to the detriment of embedded computing or processing functions. In this case, exchanges will have minimum confidentiality, the data flow between the reader and the tag will be relatively small and the data processing carried out on the tag will be very simple, if any. On the other hand, remote reading of the tag without awareness of the owner will be particularly easy when tag range is large and its processing capabilities minimum.

However, smart cards, suitable for the handling of more confidential data, will give priority to computation and processing in order to perform complex security tasks such as : cryptography calculation to authenticate the holder and protect data, secure storage of sensitive information in non volatile memory, use of more efficient operating system for multi-application management and downloading of new service applications. These contactless objects, on which a microcontroller is embedded, will have to collect more energy from the electromagnetic field emitted by the reader and thus operate in near field. In this case, exchanges with a very high degree of confidentiality and a higher data rate will be mandatory. However, due to operation in near field, the holder of the contactless object will have to make a voluntary gesture for the transaction to take place a few centimeters from the reader. Conversely, readers will have to be a few centimeters from the object to run a transaction.

To enable worldwide interoperability of a contactless system, ISO international standards have been published : the family of ISO 18000-x standards for RFID identification tags covering the frequencies bands of 125 kHz, 13.56 MHz, 433 MHz, UHF(868-915MHz), 2.45 GHz and the ISO 14443-x standards for contactless proximity cards at 13.56 MHz and ISO 15693-x standards for contactless vicinity cards, also at 13.56 MHz. These standards describe the air interface part of the communication channel and thus allow any reader to decode information transmitted by a contactless object and vice versa. To decode doesn't mean to decipher,

of course, and covert transaction are possible and manageable at transport or application layers. To ensure reliability of exchanges and prevent interpretation of content, data can be encrypted by means of a secret shared by the card and the reader.
Standard ISO 14443 also describes frame formats and collision management : a deterministic approach for type A and a probabilistic approach for type B, associated with the communication protocol and specific commands for contactless cards.

Contactless cards versus contact cards differences

Comparing contactless cards with contact cards, three main differences can be pointed out :

1. The use of a RF channel
2. The remote power transmission
3. The multi card identification protocol

RF Channel : The use of a RF channel whatever the operational range or standard enables new possibilities of spying the communication over the air between the reader and the card. The power transmission to supply the card allows anybody with a reader to start a transaction with a passive card without the awareness of the owner of the card. Because several cards can be together in the electromagnetic field of the reader, the protocol has to manage a specific initial phase of the protocol in order to inventory the full set of cards that are in the field. This is the so-called singulation phase which is performed before the authentication phase.

RFID : Smart Cards and Electronic Tags

To day, most contactless devices are used in Radio Frequency Identification (RFID) applications. These RFID devices can be split into two main families :
– The identification cards, mainly devoted to identification of a subscriber to a service.
– The electronic tags, to identify items for inventory or supply chain management
If we compare RFID smart cards versus RFID Tags we can notice the basic differences : Cards are used by people that are in most case only one in front of the reader and can cooperate and detect threats. They can perform some MIPS for processing. Tags are used for objects traceability. Object are inert and without any threats detection. Chips have no MIPS computation capabilities.

Low Resources Devices and Security Issues

RFID contactless devices support some very basic characteristics that bound drastically the security protections :

1. Passive devices : portable contactless objects are basically passive device and so don't support any embedded energy source. A first consequence is that a power on/off switch is useless and there nothing to do for the user to start a transaction when the device is in an electromagnetic field.
2. Low Power device : the supply power which can be collected by a contactless device is very low or even ultra low. Basically there are two main families : a) smart cards that use the near field for a magnetic coupling and can collect roughly some few tens of mW at some centimeters (2 to 10) and b) electronic tags that use UHF fields and propagation mode and backscattering and can collect some few tens of μW at long range (3 to 7 meters). The result is that the power budget is too drastic for the chip to support complex security functions and embedded protections, if any, are very limited. So far, basic electronic tags don't support any protection.

3. Low Resources devices : Because of low power supply budget and low cost chip constraints the embedded computational power is not so large on smart card chips and allows roughly, in the best cases, bloc ciphering such as DES, TDES, AES, RSA and ECC. Basic electronic tag chips for items traceability don't support any encryption processing.
4. Low Cost Devices : The area budget on the surface of the die is hardly limited due to cost constraint. Security functions must be cheap. But we can notice that Moore's law helps to increase security functions at least for the digital ones.
5. One Reader and Many Cards or Tags Identification protocol (Anti-collision) : Contrary to contact card systems where there is only one card connected to the reader, several cards may be into the electromagnetic reader field and collisions of messages can occur if not prevented. To tackle this problem, the first phase of the protocol is to identify all the present cards and to select only one to perform the transaction, which is the so called singulation phase.

To conclude the low resources aspect of contactless devices, it is worth to mention that the reader enjoys in most case relatively large power supply and so can run high processing capabilities to manage digital receiver and cryptographic functions. We must to keep in mind that the contactless system is fundamentally an un-balanced one.

Security of systems

To guarantee the security of a transactional system, four fundamental characteristics must be verified :

1. Confidentiality of data : A secure transaction must kept the data covert. No external spy may have the opportunity to get the exchanged information in clear text. To address this problem, digital cryptography is today the right answer. Designers have to check that there no possibilities of side channel leakage that can allow to hackers to get the information in an effective way.
2. Integrity of data : The system must ensure that the received data are strictly identical to the emitted data
3. Authentication of parties : Basically there are two parties : the reader and the contactless device. Before starting a transaction the two parties have to authenticate each other (mutual authentication) in order to be sure that one of the parties is not a fake or a clone.
4. Disponibility of system : a secure system must provide the service in any situation. Service must be correctly completed or if not possible, aborted in a secure state. Denial of service is a new emerging threat in information technology domain. Here it is worth to point out that some protections that are proposed for contactless systems can be easily turned out into devices producing denial of service (see below, physical attacks chapter).

Typology of threats and attacks

Three main classes of attacks can be identified on contactless systems closely related to the specificity of the contactless RF link :

1. Passive listening
2. Remote activation
3. Denial of service

Other classes of attacks, which are not directly connected to the contactless RF channel, are considered outside the scope of this paper.

Passive listening : This is the classical "communications interception" that is currently used for wireless systems. It exploits the RF channel that emits information over the air from both reader and card. It is worth mentioning here that, due to the intimate unbalanced nature of the two emitters, it is basically easier to listen the reader than the card. Radiofrequency channel attacks (RFA) belongs to that class of passive attacks, but are not oriented on exchanged data messages but rather on observation of the computation activity of the chip. Remote activation : This is a very specific possibility of contactless systems. A malicious reader can remotely supply and start a card transaction, because no mechanical or electrical action is required except to radiate a modulated harmonic magnetic field. So a fraudulent reader can trigger a transaction easily. Glitches of power, frequency or phase transmitted through the RF channel and dedicated to fault injection belong also to this class. Denial of service : Because the contactless link is accessible by anyone it is possible and very easy to disturb the communication with jammers or malicious protocols and to prevent the system working. With high power electromagnetic fields it is also possible to destroy some RF parts of the systems close to the antennas. Some malicious scenarios can carry out attacks that are a combination of these previously described classes. For instance, relay attacks or man in the middle attacks are combinations of the first two. If we examine the results of these attacks, it is also important to notice that the first class of attacks doesn't disturb the systems and create no immediate loss to the user or gain to the attacker. However it could be a preliminary phase of a more complex attack. The second class allows immediate potential gain or loss. They can leave the system undisturbed or not, according to the technique used to perform the attack. The third class of attacks doesn't produce direct gain for the attacker, but mainly losses for the operator or the user.

From the basic nature of the RF link, another very important distinction must be done about the location where the attack takes place. Basically we can identify to main areas : on one hand in a private room such as a building, a laboratory or at home and on the other hand in a public place such as a street or a commercial center. Of course, expected results and difficulties are not the same. Laboratory : anybody can do anything up to the extent of his budget. Experimentations are easier but direct gain is not evident except for preliminary phase of a more complex attack. Public place (or "street attack") : gains are more important but the difficulty to perform the scenario may turn out the attack to be very uneasy to be carried out.

Economical and Societal context and trusted computing

The main debate on RFID at the societal level is security versus privacy. Many companies that deploy contactless systems are profit companies and fraud is a severe competitor. Security is basically promoted by operators for evident reason of preventing economical losses or unauthorized accesses to subscription-based services. In some commercial or economical aspects it is also mandatory to protect the users. The targeted asset of attacks is the business. Fraudulent attacks on the contactless technology have to be nullified. Privacy is only a matter of concerns for users that will live more and more in a digital world with one or several digital doubles. Here the asset is individual freedoms and more specifically in this case the protection of digital personal data. With the trumpeted arrival of Ambient Intelligence, digital data privacy is a key point. Several contactless low resources smart devices have shown some connections with privacy concerns :

- RFID electronic tags for items identification in supply chain at retail stores.
- e-Citizen contactless cards (e-passport, e-identity,...) for person identification

For the daily usage of this emerging and rapidly spreading technology, some questions arise from the basic user, such as :
– Do I communicate with the right reader ?
– Is somebody spying my transaction ?
– Is somebody trying to communicate with my passive device ?
– What are the data exchanged over the air ?
Here, it should be pointed out that not only the users but operators that deploy the system ask for quite the same questions. The former want to protect his privacy, the latter to protect his business. In both case they need to trust the trusted computing supported by these contactless portable objects and must understand the underlying security in order to trust it.

RFID electronic tags : The concept of ambient intelligence (or AmI) is a vision where humans are surrounded by computing and networking technology unobtrusively embedded in their surroundings. The privacy concerns have been raised very early with the introduction of RFID smart devices. In particular the introduction and potential generalization of electronic tags on mass markets products and at the retails level at every shop or supermarket have enlarged the potentiality of traceability of persons. The interoperability of automatic data capture systems and the combination of tagged items carried by a person may be used to trace and to locate this person when moving in commercial center or urban areas. e-Citizen contactless cards : Concerns are also very high with the use of contactless e-passport. Some hostile distant reader may check your nationality and trigger attacks. A same scheme may be ruled out with an identity card. High sensitive privacy personal data may be potentially downloaded from contactless health cards without the agreement of the owner.

Among the basic fears that users, consumers and citizens raise when addressing the contactless technology, one can mention :

1. A human person cannot sense the electromagnetic field
2. The identification code is unique. Comparing with the bar code, the RFID code includes the serial number of the items that enables total traceability. Groups of electronic identifiers (RFID) turn out to be a signature of a person and so to be traced easily.
3. RFID readers are small enough to be embedded in a cell phone.
4. The localization of person is possible when she is close to a reader
5. Citizens and consumers do not have access to the history of the readings.

In that "privacy situation" the main drawback of these contactless systems is that there is no switch on/switch off mechanism to assess the will of the owner to use them. They can be remotely and covertly activated and triggered without the awareness of the owner. So trust starts to shift down. Privacy is the ability of an individual or group to keep their lives and personal affairs out of public view, or to control the flow of information about him. It is clear that AmI may endanger the privacy of citizens. On the other hand, AmI aims to offer a lot of on-line services that anybody can get only if he identifies himself or the items he is dealing with. So is the difficulty.

Public regulation

Government and international institutions aim to improve the protection of privacy of citizens through attempts of regulations. The main elements proposed by the European Commission in a recent recommendation document can be summarised as follows :
– Operators should conduct a risk assessment prior to deploying an RFID application to ensure privacy risks have been properly evaluated.

- The industry is encouraged to establish codes of conduct governing the use of RFID. These codes can be sector-specific to reflect the different ways operators use technology.
- Operators should provide a minimum level of information to users of tagged products, such as the purpose of the application or the identity of the operator.
- Operators should follow a series of information security measures when deploying RFID applications.
- Specific provisions should apply to the retail sector when a product containing an RFID tag is sold. The Commission is considering a harmonised sign that would inform consumers when RFID tags are used. In addition, to guarantee consumer choice and control, RFID tags that contain personal data should be automatically deactivated at the "point of sale" unless the consumer decides otherwise.
- Member States are encouraged to raise awareness among citizens and SMEs through information campaigns or large-scale pilot projects.

Consumers associations

Consumers associations were among the first organizations to react to privacy concerns about RFID tags. Consumers Against Supermarket Privacy Invasion and Numbering (CASPIAN) is a consumer group fighting retail surveillance schemes since 1999. With thousands of members in all 50 U.S. states and over 30 countries worldwide, CASPIAN seeks to educate consumers about marketing strategies that invade their privacy and to encourage privacy-conscious shopping habits across the retail spectrum.

Public organization

CNIL is a French public organization devoted to privacy protection of citizens against the emerging of new information and communication technologies. Of course RFID is a main concern. The technology of radio-identification (RFID) becomes a major economic stake in particular in the applications of the distribution and transport. Because of their massive dissemination, individual nature of the identifiers of each marked object, of their invisible nature, and risks of shaping of the individuals, the CNIL considers that RFIDs are personal identifiers within the meaning of the law "Informatique et Libertés". "Der Bundesbeauftragte für den Datenschutz" in Germany, "the office of information Commissioner" in UK and "Garante per la protezione dei dati personali" in Italy are other European governmental organizations that manage privacy rights.

Threats and scenarios

Risk analysis

Risk analysis must be performed upon contactless systems on two main topics : a) prevention of economical fraud and b) protection of privacy. The targeted assets and motivation of attackers are not quite the same. Possibly some countermeasures could help both. Anyway in both cases threats and vulnerability must be addressed, focusing on the contactless RF interface. Risk analysis on contact chip is nowadays a stable and very accurate process. Among vulnerability of the contactless interface, it is important to list :

- Bidirectional data communications over the air
- Unidirectional power transfer over the air

– Clock transfer over the air
– Passive devices and no on/off switch
– Load based retro modulation
– Singulation protocol Misuse of critical commands (ie : kill command)

Among threats on the contactless interface it is worth to mention :
– Substitution of reader and/or transponder (clone, relay, man in the middle,...)
– Jamming of RF channels
– Eavesdropping of RF channels
– Abuse of the reader or the card
– Disclosure of secret keys or personal data
– Destruction of the RF communication link
– Tracking, localization with (skimming or eavesdropping)

Countermeasures are the key point of risk management and are devoted to nullify or to reduce as far as possible the risk that a threat jeopardizes vulnerability.

Scenarios analysis

Attack scenarios on contactless devices must be investigated in a very careful approach. .One important work is to check if the attack is directly connected to the contactless link or not. If yes, attack scenarios must be described in a common framework. For example the following characteristics must be describers :
– Attacker type
– Motivation of attacker
– Attacked asset
– Attack methodology
– Feasibility of attack
– Cost of attack
– Etc...

It is worth to quantify the attack in order to address first the more relevant ones. A computable risk factor could help to do that. It is also important to point out that combined attacks have to be carefully investigated.

Application domains and products

Four main families of products and associated use cases are involved in contactless technologies :

1. Smart cards
2. Electronic tags
3. NFC devices
4. Medical or industrial implants

Nowadays, the following industrial applications are using contactless smart card technology :
– Payment card (credit/debit, electronic purse)
– Electronic passports
– Digital identity cards for citizens (vehicles, etc.)
– Multi-purpose cards for the citizen, including mass transport and banking
– Various licenses issued by a government authority (driving license, residence license, work license, etc.)
– Restricted area access control

NFC, a short-range communication technology based on ISO14443 and ISO 15693 standards, is now embedded into mobile device environment. NFC applications are still in an early phase although there have been some real implementations as the transport ticketing trial in several European countries. Another area, such as automatic object identification by electronic tags (RFID tags) is rapidly expanding. Specific threats have been identified with respect to contactless technology and must be compared with smart card products.

Typical features of contactless scenarios

Sequences of basic attacks : It is worth to mention that a scenario of attack on the contacless RF interface may be a sequence of basic attacks such as RFA followed with skimming, in order to hack a card owner and the related service operator. Laboratory attacks versus outdoor attacks : Because of the basic properties of the RF contactless channel and its over the air aspect, it is possible to perform outdoor attacks. These attacks are fundamentally harder to perform outdoor than in the laboratory, but the gain may be immediate and substantial. The best example is the relay attack, that is useless on a laboratory test bench, but attractive in the street.

Basic attacks on contactless RF channel

Before detailing the main known attacks on the contactless channel, it is worth trying to globally structure them from a tutorial point of view. A first aspect can be the goal of the attack. Two goals can be identified : malicious or fraudulent use or prevention of operation of the system. A second aspect is about the way the RF channel is attacked : in a passive way that means no EM field is emitted by the attacker or in an active way when a EM powering field is used. A last one is about the state of the system after the attack. It is still operational if the attack is reversible and no more operational if not. Finally it is worth to mention what is the gain for the attacker. The following table attempts to summarize the whole set of known contactless RF channel attacks.

Global goal	EM mode	Syst. Integrity	Attacker gain	Attack type
Fraudulent	Passive	Reversible	Data	Eavesdropping
Fraudulent	Passive	Reversible	Keys	RFA
Fraudulent	Active	Reversible	Data	Skimming
Fraudulent	Active	Reversible	Service	Relay
Fraudulent	Active	Reversible	Service	ManInMiddle
Fraudulent	Active	Reversible	Bypass check	Fault injection
Denial	Passive	Reversible	Denial service	Blocker
Denial	Active	Reversible	Denial service	Jamming
Denial	Active	Ireversible	Denial service	Misuse of kill
Denial	Active	Ireversible	Denial service	Destruction

Of course other attacks exist and must be mentioned such as :

- Combined attacks on contact and contacless modes
- Introduction of viruses
- Mechanical or chemical destruction of the antenna, the substate or the chip.

But because they are not directly connected to the RF channel they will not be consided on this panorama devoted to secure contactless technology.

Eavesdropping

In this case a hacker discreetly listens to a running transaction in order to attempt to retrieve information that he could use in a fraudulent manner at a later date. This passive attack consists in listening transactions between an authentic reader and an authentic contactless card and trying to collect data or information. In that situation, the owner of the contactless device is normally aware that a transaction is being carried out. Basically data are the contents of the exchanged messages and can be easily recovered if they are not ciphered. It is the case of electronic RFID tags for example. Another information can be collected by registering the sequence of exchanged messages that can tell us which kind of transaction has been achieved. Most contactless protocols can be accessed through public documents, such as description of standard devices. Data emitted by the card as well as data emitted by the reader can be eavesdropped. Because only the latter one emits an electromagnetic field, it is easier to intercept its broadcasted data messages. Basically the messages emitted by the reader can be listened at a larger distance than those emitted by the contactless device. The eavesdropping operational distance depends upon the sensitivity of the receiver and of the receiving antenna.

RFA

In the world of contact cards, a well known class of attacks named single power analysis (SPA), differential power analysis (DPA) or electromagnetic analysis (EMA) enables the hackers to recover the secret key used by the cryptographic processor of the card. This class belongs to observation attacks. The hacker registers passively the power consumption variations of the chip, directly or through electromagnetic probes, before running some computation to guess the right key. This classical side channel attack can be transposed to the contactless world. Moreover the technique used to modulate the return link can be helpful. The retro modulator is based on the variation of the load of the chip, so equivalently on its power consumption. Because power consumption is basically representative of the current computation work of the chip, the chip activity directly modulates the embedded emitter. Thus power consumption activity of the chip is directly transmitted through the RF channel. Finally a hacker can register the modulated signal and recover the key, especially if the SPA attack is sufficient to succeed in, such as for RSA exponentiation.

Skimming

Skimming is an active attack. Because a contactless card has no ON/OFF switch, it is not necessary to ask for authorization from the card holder via a voluntary gesture on his/her part to activate this card. In this case an unauthorized reader may try to take control of the card. Simple authentication of the reader by the card may solve the problem, except if the attacker has previously got the password or the key. The hacker used a specific reader of his own to trigger a communication with the targeted card. Because the card is a passive one, it reacts to the ambient RF field and starts the communication protocol. According to the used system, more or less information can be recovered by the attacker. e-passport could be an ideal target of that kind of attack. If not prevented, it is possible to remotely get sensitive information on a person. Of course some access control functions have been embedded to manage the trouble. Because a human being is not able to sense an electromagnetic field, this kind of attack is very worrying for privacy protection.

Relay

Relay attack is a more serious one because of its basic simplicity. The objective is to trigger a normal transaction between a true reader and a true card with a purely transparent relay of communication. The relay is made of a fake card and a fake reader that are connected together with any communication link, fast enough to respect the timeouts of the protocol. Of course it works because the two linked true contactless devices were designed to work together. With such an attack, it is possible to substitute a fake contactless device with a true one which can be very far from the true reader. Relay attacks involve two different devices and as a consequence two attackers that should coordinate each other except if the relay is really short. The relay attack is based on a specific weakness of the contactless smartcards or RFID tags that is the possibility to activate the device without the consent of the user. A user is not able to switch off his card, and thus an attacker can, therefore, access the card discreetly, without knowledge of its owner, and relay information through a communication link between the card and a remote reader. The reader will assume that the card, and by implication the user, is in close vicinity. Using this attack on cryptographic authentication schemes, the attacker would be able to convince both reader and card that they share a common secret key.

Man in the Middle

"Man in the middle" attack is an over set of relay attack. It is indeed quite similar but with the distinctive feature that in this attack the bit streams that flow into the relay can be modified. Since long range relays required demodulation and decoding of data, it is not a problem to change some bits. This additional processing may need some more time but it could be shorter than the timeout of the protocol. When the communication is ciphered, the attacker has to know the secret key and the algorithm to change the data.

Fault Injection

The basic idea here is to use the RF channel as an entrance door into the chip. It is possible to modify the power, the frequency or the phase of the carrier in order to inject an operating fault during the microcontroller code execution in order to disturb the execution of the program and prevent a control to work and try to get confidential information.

Blockers

With blockers, a new class of attacks is presented : attacks that aim at preventing the right operation of the system. Blockers use the basic protocol in such a way that the transaction doesn't progress any more and loops endlessly in a preliminary identification phase. Anti-collision phase of the protocol is a nice place to do that. At that stage of the protocol, the reader tries to identify all the tags that are present in the electromagnetic field. To complete its inventory the reader checks if there is still a remaining tag. It is possible to abuse the reader with a device that doesn't go silent when the reader said it is identified, or that emulates all the possible identification codes or that triggers continuously collisions.

Jamming

With very basic equipments, it is possible to prevent a reader or a tag to work with jammers. The idea is to emit a noisy signal in the same bandwidth as the reader or the tag RF channel in order to blur the communication. Only few watts are required to do that.

Destruction

This attack consists in destroying some part of the contactless system in order to prevent any future operation in an irreversible way. Well known destructive attacks through the RF channel are :

1. High power RF field. Highpower RF pulse emission and over voltage at the pads of the antenna or the chip. According to the current RFID standards, contactless cards must resist to a high electromagnetic up to a given threshold. Beyond this threshold the manufacturer doesn't and cannot guarantee the reliability of the product.
2. Misuse of the kill command. Some electronic tags such as ePC products support the kill command, the purpose of which is to inhibit definitively the tag for any further operation in order to preserve privacy. Malicious use of that possibility may endanger tags which is no so difficult because the command is only protected with a 32bit password.

Malicious use of counter-measures : denial of service

Some counter-measures proposed (see next chapter) and available to anybody may be turned out towards malicious usages to generate a denial of service. The blocker is a good example. Indeed, while perfectly complying with relevant standards, this approach prevents proper operation of the system by simple emulation of a multitude of tags and saturation of identification time.

Protections against physical attacks

Many solutions have been proposed, essentially from research laboratories. Among them we may point out :

Main purpose	Vulnerability	Attack	CM examples	Asset
Privacy	OTA Data channel	Eavesdropping	Ciphering, noisy RF channel	Data
Fraud	Load based Retro modulation	RFA	Chip Design	Keys
Privacy	Passive device	Skimming	Faraday, CPM	Data
Fraud	Passive device	Relay	Faraday, CPM	Service
Fraud	Passive device	ManInMiddle	Faraday, CPM	Service
Fraud	OTA power channel	Fault injection	Chip Design	Bypass check
Fraud	Singulation protocol	Blocker	Reader protocol monitoring	Denial service
Fraud	RF channel	Jamming	Reader detection	Denial service
Fraud	Protocol	Misuse kill	More crypto	Denial service
Fraud	RF channel	Destruction	Faraday, chip design	Denial service

CM : Countermeasure
OTA : Over the air
CPM : Contactless Privacy Manager
RFA : RF Analysis

Ciphering of data The basic solution to protect of confidentiality of data is to cipher them with cryptographic solutions. This technique is well developed and perfectly mastered. The only drawback is that it is resources consuming in power and area on the chip and that it requires keys management. In conclusion this solution is right for complex chip such as smart card ones, but it is today irrelevant for low cost tag chips.

Noisy Readers, noisy tags Possible protections focus on the noisy tag and/or noisy reader concept, mainly for low resources devices such as tags. This approach aims to increase the confidentiality of exchanged data between the reader and the tags and to protect privacy. Protocols based on this concept can be used also by a reader and a tag to exchange a secret. These concepts do not impact the current air interface standards. However both assume that the readers are trusted.

Basically, noisy readers act like jammers to decrease the signal to noise ration of the contactless return link (tag to reader). Noise is added to the carrier when the tag is emitting data. But because readers know the noise they are emitting, some signal processing can easily cancel it from the received signal and so decode the tag message. Other close readers are blurred by the noise and are unable to remove it. This technique can be used at the tag level, but because they don't emit electromagnetic power, the masking effect doesn't spread on a large volume around the tag and the method is not so efficient.

Noisy tag technique is a quite similar protocol that can be used between an RFID tag and a reader to exchange a secret while preventing eavesdropping. Special tags, called noisy tags, emit a pseudo noisy signal in the RF channel when the tag is emitting messages. Because the reader knows the pseudo sequence emitted by the noisy tags, it can remove it from the received signal and decode the tag message.

Faraday cage The well known purpose of the Faraday cage is to stop electromagnetic field. The consequence is that the contactless device which is inside the cage, doesn't receive neither power nor data to be started. Faraday cage is used as a very low cost shield solution to prevent any hostile reader to access the passive contactless device without the consent of the owner. Typically, e-passport are using that kind of solution for privacy protection.

Blocker tags The basic principle of blocker tags is to exploit a very specific characteristic of the RFID protocol which is the singulation phase. This phase occurs at the early beginning of the transaction and aims to identify all the contactless devices that are present in the electromagnetic field of the reader . The blocker tag works by always responding to the inventory query of the reader and by emulating all the identification codes set or by creating collisions continuously. In this way, it can passively jam the reader, forcing the reader to endlessly inventorying very large sets of tags and so never identifying the tags that are in the electromagnetic field. This protection may be also assimilated to denial of service. However a fine reader can detect malicious use of blocker tags by checking the too large number of expected tags (over 1000 for example) or by checking the inconsistency of some ID numbers with the subset that it is expected to be read.

Contactless Privacy Manager Basically, the Contactless Privacy Manager (CPM) security module is a standalone and a personal device. Its main purpose is to give or increase confidence in what the user trusts. Another possibility is to incorporate some of its security functions both on the reader and on the portable device. However for RFID electronic tags, the standalone solution looks better, from an economical point of view.

The ideal standalone CPM must integrate the following functions : a) a sensing of the contactless environment , in order to detect all known contactless standard activities (smart cards and RFID tags) and be able to warn the user in real time about potential tentatives from hostile readers to try to communicate with his own devices, b) a management of black or white lists for sorting trusted readers or trusted tags from hostile ones, c) a basic communication controller able to control every data exchange between personal devices and readers, d) a communication jamming mechanism enabling blocking unwanted communications and performing counter measures against eavesdropping, e) an identity manager running basic privacy

protection functions such as ID control, anonymity management and personal data management, f) an authentication manager enabling authentication primitives between readers and low resources devices (RFID tags) where authentication is not supported and authentication of the surrounding readers, g) a friendly user interface able to display on line every messages and control of every data exchange, h) an audit memory for recording and checking of every transaction for off line analysis, i) a destruction function able to manage CPM auto destruction or contactless device destruction or kill function.

Some laboratory works have been done in this way such as the RFID Guardian or the CPM.

Tag destruction and industrially proposed countermeasures

For electronic tags, the main purpose of countermeasures dedicated to the RF channel is to protect privacy after the consumer has bought the products, while allowing the use of RDID systems along the supply chain. They are used at the industrial level.

Responding to RFID privacy concerns, IBM has developed a "clipped tag" technology, offering consumers the ability to tear or scratch off RFID antennae, eliminating the threat of an unauthorized reading of the tag.

ePC has created a standard of electronic tags that includes a "kill" command to make RFID tags inoperable altogether. But that action, once taken, cannot be reversed. This command is weakly protected by a password. Destruction of tag would present a problem when consumers return items to the store – the RFID tag could not be read at all. With the clipped tag technology, though the antennae would be inoperable, the tag could still be read if held very close to a RFID reader, according IBM. This process shifts the choice of privacy management in the hands of consumers. It provides visual evidence that the tags have been modified, psychologically important to those concerned about RFID privacy, but does not totally destroy the tag.

According to Verayo, this American company has just released the first un-clonable RFID tag. A specific Physical Unclonable Function (PUF) technology, developed at MIT, has been introduced in the design of the chip. This authentication technology provides an individual signature for every chip, similar to fingerprints.

A U.K. firm has recently developed an on/off "switch" for RFID cards that could protect cardholders from being hacked. The cardholder activates the RFID transmission by squeezing the card between his thumb and forefinger when it must be scanned by a reader.

Recent developments at CEA-LETI

In order to tackle fraudulent and privacy issues on contactless devices, some new technological development have been investigated at CEA-LETI. In particular one can focuses on the following :

1. Noisy readers : In order to prevent eavesdropping attacks against the data RF channel for low resources devices, the reader emits a known noisy signal with is added to the carrier. The reader can perform signal processing to cancel this known signal, while eavesdropping readers are unable to extract the data signal. This technique can also defeat some RFA attacks. There is no modification to be done on the contactless devices.
2. Secure stream ciphers : Stream ciphers may be the right candidates for low resources devices. Current investigation works are devoted to best tradeoff algorithm selection and mainly on the secure design of the low power cipher bloc. Security tests are performed for validation.
3. Contactless Privacy Manager (CPM) : CPM is a set of secure functions dedicated to privacy enhanced technology. The main functions are : a) RF field sensing to early detect

rogue readers, b) management of black lists of tags, cards or readers, c) control of communication and d) monitoring of transmitted personal data. Functions can be grouped in a specific nomadic device that looks like and acts like a man-in-the middle system, or can be embedded in readers and cards or tags.

4. Embedded micro-battery : specific above-IC technological developments have been achieved in order to built micro-batteries on the silicon chip. Few energy is available, but it is sufficient to power intrusion detection and erase of sensitive data.

Conclusions

In this paper we have introduced the potential threats that are possible on the contactless technology. This range of devices is basically low resources devices, turning out protection designs to a tremendous challenge. The attack scenarios have been analyzed and finally, some new avenues for protections have been presented. Future works will focus mainly on prevention of denial of service and relay attacks which are still to be investigated and for which no industrial solutions are available. The final goal aims to maintain or increase trust and confidence of users and operators in new emerging technological solutions.

Références

1. Federal Office for Information Security, Security Aspects and Prospective Applications of RFID Systems , Germany 2004
2. Z. Kfir and A. Wool, Picking Virtual Pockets using relay Attacks on Contactless Smartcard Systems , Tel Aviv University, 2004
3. A. Juels et al., Security and Privacy Issues in e-passport , 2005
4. G. Hencke and M ; Kuhn, An RFID Distance Bounding Protocol, University of Cambridge, 2005
5. A. Juels et al., The Blocker Tag : Selective Blocking of RFID Tags for Consumer Privacy, CCS'03, 2003
6. G. Hancke, A practical relay Attack on ISO 14443 Proximity Cards, University of Cambridge, 2004
7. G. Ko and P. Karger, Preventing Security and Privacy Attacks on Machine readable Travel Documents, Univ. Columbia and IBM, 2004
8. B. Schneier, Fatal flaw weakens RFID passports, Wired News 2005
9. Doo Ho Choi at al. Privacy Protection for Secure Mobile RFID Service, S&P IEEE 2006
10. Ari Juels, Ronald L Rivest, Michael Szydlo, The Blocker Tag : Selective Blocking of RFID Tags for Consumer Privacy, CCS.03, October 2003.
11. Stephen A. Weis, Sanjay E. Sarma, Ronald L. Rivest, and Daniel W. Engels, Security and Privacy Aspects of Low-Cost Radio Freguency IdentiPcation Systems, First International Conference on Security in Pervasive Computing(SPC) 2003.
12. K. Fishin et al, Some Methods for Privacy in RFID Communication IBM ESAS 2004
13. Melanie R. Rieback et al., A Platform for RFID Security and Privacy Administration, Vrije Universiteit, 2006

14. C. Castelluccia and G. Avoine. Noisy Tags : A Pretty Good Key Exchange Protocol for RFID Tags. In J. Domingo-Ferrer, J. Posegga, and D. Schreckling, editors, Smart Card Research and Applications, CARDIS 2006. Springer-Verlag. To appear.

15. C. Castelluccia and P. Mutaf. Shake Them Up (A movement-based pairing protocol for CPU-constrained devices)! ACM/Usenix Mobisys, June 2005, Seattle, USA.

16. F. Vacherand et al. New Technologies for Contactless Air Interfaces. e-Smart'05, Sophia Antipolis France, 2005

17. F. Vacherand, New Technologies for RFID. sOc-EUSAI'05 Grenoble France 2005.

18. F. Vacherand, Security and Privacy for Contactless Devices, e-Smart'08, Sophia Antipolis France, 2008

19. Garfinkel, S. L., Juels A., Pappu R. :RFID Privacy : An Overview of Problems and Proposed Solutions, IEEE Security and Privacy, May/June 2005, pp34–43.

20. Ilan KIRSCHENBAUM, Avishai WOOL, School of electrical Engeneering, Tel Aviv University : How to build a Low-cost, extended-range RFID Skimme, may 8th 2006

21. urosmart : White paper RFID technology security concerns, understanding secure contactless device versus RFID, October 2007.

EMAN : Un cheval de Troie dans une carte à puce

Jean-Louis Lanet, Emilie Faugeron, and Anthony Dessiatnikoff

XLIM-UMR CNRS 6172,
83 rue d'Isle, 87000 LIMOGES, FRANCE
jean-louis.lanet@unilim.fr, emilie.faugeron@etu.unilim.fr,
Anthony.Dessiatnikoff@etu.unilim.fr

Les cartes à puce sont un élément de base dans la confiance que nous pouvons avoir dans les systèmes électroniques et les transactions sécurisées. Néanmoins depuis quelques années certaines attaques ont pu être réalisées permettant d'obtenir des informations secrètes. Ces attaques sont essentiellement effectuées à l'aide de fuites d'information obtenue par des écoutes passives du comportement des composants électroniques. La possibilité d'ouvrir les cartes à puce au monde externe par chargement de code après la fabrication peut amener de nouvelles attaques que nous qualifierons de logiques. Ce type d'attaque est basé sur l'exploitation de faille de spécification ou plus généralement d'implémentation. Très récemment (e-smart 2008) l'Université de Nimègue a montré la possibilité de générer des attaques traditionnelles de buffer over flow ou de confusion de type en exploitant des caractéristiques de la spécification Java Card. Les hypothèses utilisées dans cette attaque sont l'absence de vérifieur de byte code embarqué et la possession des clés de chargement.

Nous présenterons ici une nouvelle attaque : EMAN[1] qui utilise une faiblesse de la spécification Java Card. Cette attaque permet de charger un cheval de Troie dans la carte sous les mêmes hypothèses que l'Université de Nimègue. Une fois le cheval de Troie en place il est possible de modifier le code d'une application tierce pour laquelle normalement le firewall intégré dans la carte doit refuser l'accès. Nos travaux portent actuellement sur la combinaison de l'attaque proposée par Nimègue et l'attaque EMAN afin de relaxer l'hypothèse sur la présence du vérifieur de byte code.

Nous montrons que cette attaque fonctionne facilement sur des cartes anciennes et nécessite un peu de savoir faire pour contrecarrer les contre mesures intégrées sur des cartes plus récentes. L'arrivée de la norme Java Card 3.0 connected edition par contre ne permet plus ce type d'attaque de par le choix effectué sur le format d'entrée.

[1] Cette attaque a été mise en œuvre par des étudiants de l'Université de Limoges, www.cryptis.fr

Mobile Transactions : trust & privacy aspects

Jean Claude Paillès

ORANGE-LABS, Orange FT Group
jeanclaude.pailles@orange-ftgroup.com

Abstract This paper describes first the standardization work being done in the Mobile Phone Working Group in the TCG forum. After a few examples of use cases, it addresses the topic of privacy, which is of primary importance for the roll out of trusted computing technologies. It then describes the general objectives of the TSC project, and then the demonstrator under development by Orange-labs.

1 Differences between TPM and MTM

1.1 Reminder of main TPM functions

The work being done in the TCG forum [1] is well known. Basics of concept of trusted computing developed in this group can be summarized as follows :

Configuration control : the basic concept used here is the "secure boot process". During boot, a subset of the firmware of the mother board (BIOS) is able to load a small progarmm called "boot", which is capable of loading an OS loader, which in turn is able to load the OS, etc. Integrity measures (ie hash code) of the software loaded in each step is stored in the TPM memory, in PCR (program control register). They can be "reported" to a distant server through a signature mechanism. So the distant server may verify that the configuration is genuine (ie if it has been modified intentionally or not) and if it can'trust it. This mechanism is also referred to as "transitivity of trust" since starting from a "root of trust" (TPM+BIOS subset) trust property is extended step by step to the whole configuration.

EK : An RSA key pair called "endorsement key" EK certified by the board and TPM manufacturer and the TCG organisation. EK is the primary key which enables to set-up subsequent keys, if needed.

Secure storage : any data stored in RAM or external memory can be enciphered and/or "macced" in order to protect its confidentiality and/or integrity, the process for obtaining these data in clear being conditioned by the value of some PCRs, thus conditioned by the integrity of the OS or part of it.

TPM Commands : local or distant all commands sent to the TPM can be sent by a program in the PC or by a distant entity ; All commands use an authentication value for verifying its origin. In distant case, a secure session mechanism enables protection of sensitive parameters of each command, like the auth-value.

Locality : The TPM may detect a local signal which, by construction, can't be emulated by software, thus proving agreement of the local user. This signal can condition commands received by the TPM.

It is important to mention that these low level mechanisms need some form of standardisation, which is the goal of TCG : additionally to the benefit of rationalization, there is clearly a need to offer to distant services a mean to authenticate and verify trustworthiness of a platform , and/or to use an implementation independent command set for triggering such or such security process inside the distant PC..

1.2 Case of mobiles : Local verification of measures

TCG approach for PC is more oriented towards the capability to prove PC-platform soundness to the external world; soundness is established through secure boot process and measures, and secure reporting of the measures to a distant server which can verify it. TCG approach for mobiles (Mobile Phone Working Group : MPWG : [2]) is quite different : first, the platform security is considered to cover radio subsystem. Radio subsystem for cellular network is very sensitive because a non respect of the rules for accessing shared radio-channels would result in a denial of service for all the mobile phones in the current cell. So OTA access to the external world needs radio software soundness. This implies that soundness has to be established locally in the mobile. For this reason, reference integrity measures (RIM) have been introduced in the mobile trusted module : MTM. They are signed by an authority which public key is known by the mobile : RVAI : "root verification authority identity". For having a capability of enabling different responsible entities for defining the RIMs, a 2 level system has been defined for signature of the RIM by RIM-entity, and certification of this RIM-entity by the RVAI. The "PCR_extend" command of TPM has been replaced by a "VerifyRIMcertandExtend" command which role is to do the same than extend, and additionally to compare the PCR value to the relevant RIM after having verified the signature and the certificate of this RIM. So the secure boot process in a mobile looks like the example shown in the drawing above : it is assumed that the status of each verify&extend command is positive. The boot sequence verifies the Bios(1), then loads and triggers a loader (2) and then load and trigger the OS (3). Many variations are possible around this example. Here, use of different PCR 1, 2, 3 may be justified for a better separation between different software developers of the different programs loaded during boot sequence. And the extend commands with textual data just enable an easier monitoring of the boot process if something goes wrong.

1.3 Ownership policy, multi-stake-holder concept

Another difference between TPM and MTM comes from the ownership policy, implied by the mobiles phones. Ownership in the sense of TCG is classical, ie it gives administration right to one entity. This entity is able through the TPM commands to set-up new keys and certificates. Consequently, it will have total freedom to make any change in the software configuration, and enable others entities to control this platform through integrity measures and reporting function... In case of mobile phones, several entities with different stakes are considered : the radio subsystem has to be controlled by a trusted entity like Manufacturer or Mobile operator. The subsystem containing operator services, message handling, browser, etc could be controlled for example by the MNO. A subsystem containing games could be under direct responsibility of the user. So, TCG MPWG has defined for a trusted mobile platform a "multi-engine" architecture : the basic engine which contains the radio subsystem is controlled by the manufacturer (or MNO[1]). Other subsystems for concepts uniformity, are also seen as

[1] This responsibility would imply for a MNO the knowledge of hundred's of mobile phone characteristics and measures, which seems to be more appropriate for mobile manufacturers.

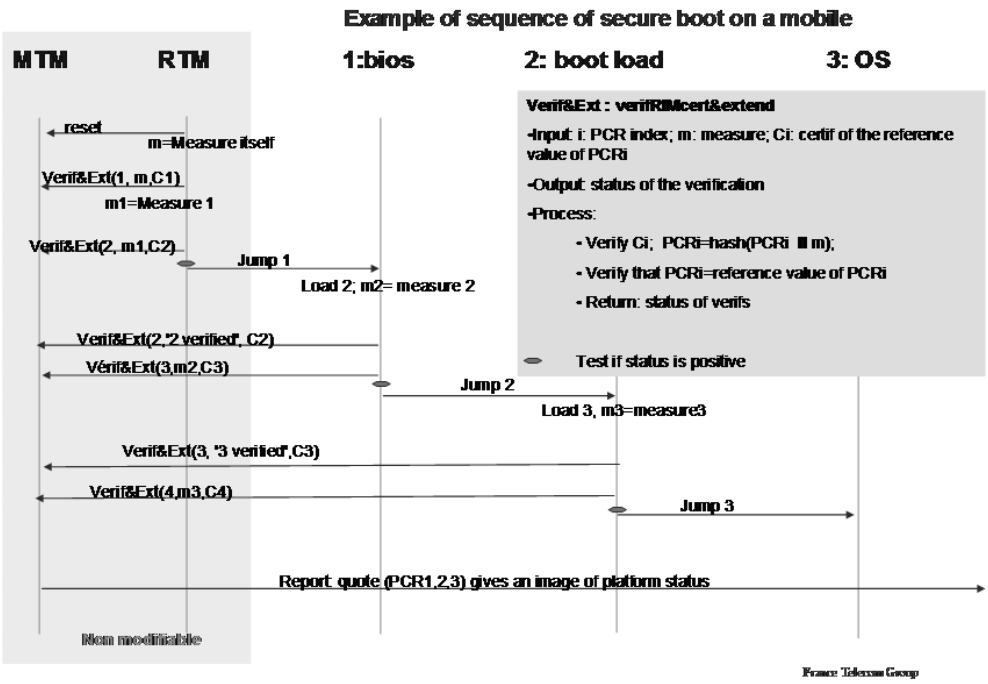
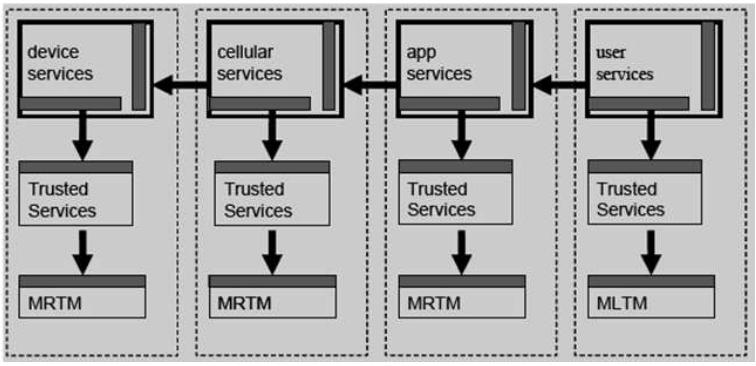


FIG. 1: Example of sequence of secure boot on a mobile.

engines, with their own capabilities for measuring, trusted boot and reporting. They rely on the services offered by the basic engine, in a kind of recursive architecture. An upper level engine relies on the services offered by a lower level engine which may be the basic engine (on the left of the drawing below). Multiple engines acts on behalf of a different stakeholder. The engines on the drawing provide services on behalf of the entities that provide the device, cellular access, an application, and user services. The solid rectangles indicate interfaces and the solid arrows indicate dependency (the arrow pointing away from the dependant entity. In this example, the device engine provides basic platform resources, which include a user interface, a radio transmitter and receiver, Random Number Generator, the IMEI, and a SIM interface. The device engine provides its services to an engine that provides cellular services. The cellular engine provides its services to an application engine, and the application engine provides its services to the user.

1.4 Two Types of MTM : MRTM/MLTM

In each engine, conventional services have access to Trusted Services, which make measurements of the conventional services and store those measurements in a Mobile Trusted Module (MTM). The device, cellular, and application engines have a Mobile Remote-owner Trusted Module (MRTM), because those stakeholders do not have physical access to the phone and



need a secure boot process to ensure that their engines do what is needed. They need a RIM capability with the extend&verify functions enabling a locally verified trusted boot. The user engine has a Mobile Local-owner Trusted Module (MLTM), because the user does have physical access to the phone, and can load the software he wishes to execute. The MTMs , local or remote, can be trusted to report the current state of their engine, and provide evidence about the current state of the engine. The reason for having two separate types of trusted modules is their differing design objectives. A MRTM is designed to be used to implement local verification; so, it must have capabilities related to RIM initialization/update and extend&verifyRIM. RIM. Initialization / update is under control of a remote owner. A MLTM is designed to be used to support remote verification for e.g. remote attestation. It doesn't need to have these capabilities. It has simply to contain a subset of the TPM 1.2 capabilities, without RIM initialization/update and extend&verify functions. An MLTM may also be used for providing local verification under the direction of a local owner through the sealed storage of keys created under control of the local MTM owner.. This multi-engine concept doesn't seem to be really essential; it is a possibility, but it is unclear whether this concept will prevail or not : see next paragraph.

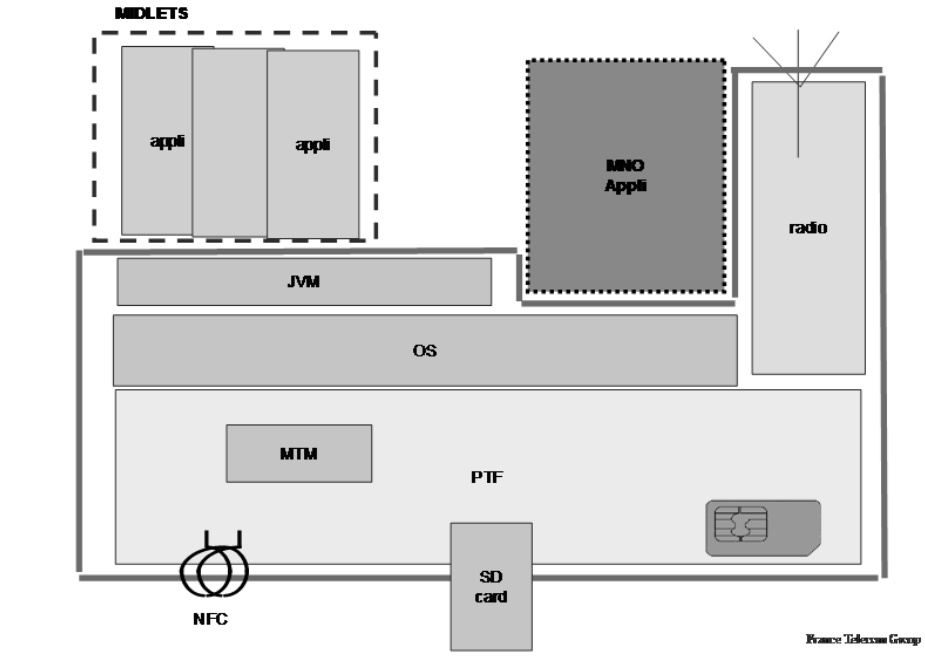
1.5 Assurance level

The TCG Mobile Working Group has not only defined specs for the MTM, but also a protection profile together with an assurance level. The required level will be EAL3, which is a rather low level. The reason why such modest requirement has been chosen is that implementation of MTM will be in a form quite different from nowadays PCs where it is a separate chip generally supplied by smart-card chip manufacturers. In case of mobile, for obvious packaging reasons, it will be generally included in the "application processor" or "base band processor" and this chip is by far too big and complex for being evaluated with a high level of assurance. Many mobile applications (see 2) will need a better assurance level. But solution will be based on a specific secure element like the SIM or secure SD cards.

1.6 Note : TCG "awareness"

The multi-engine approach seems to imply a TCG trust policy and mechanisms everywhere in the mobile. In fact several different trust policies and mechanisms may coexist, as long as an

initial state of trust is reached through a TCG secure boot, and that the others subsystems can rely on this initial state. If we consider for example the very usual case of a MIDP 2.0 mobile, with a multi-application SIM and JSR177 for interfacing midlets with applets in the SIM, we can very well stick to a mono engine approach for all the mobile hardware platform, including the JVM. A mono engine may very well be composed of different processors with their own resources in memory and peripherals, linked by some bus or serial link or shared memory. This is the case for example of many mobile phones composed of a base-band processor and an application processor.



Midlet subsystem Secure boot process in this kind of configuration guaranties the integrity of the software inside the box with solid line. OS is integer and so may be considered as trusted. The MIDP applications are apart of this architecture : the root keys for enforcing the MIDP2.0 access control policy may be provided by the SIM, and installed by the OS which is trusted. MIDLET can be considered as trusted although having no relation with MTM, because JVM is TCG trusted, roots in JVM for MIDP2.0 are set-up by a trusted part of the OS, and because MIDLET comply with the access control policy defined in MIDP2.0. This midlet subsystem is not TCG aware.

MNO applications subsystem MNO applications may be security sensitive, and so have to be trusted, but here also they can be considered as trusted even if they don't have any direct

relations with MTM, and so are not TCG-aware : this will be the case for example if the OS (considered trusted after secure boot) enable loading of some applications and control of their integrity through a signature of the MNO, which public verification key is known by the OS. These applications may be updated by the MNO which has the ability to sign the package with a version number. A control of monotony of the values of this counter has to be performed by the OS.

Distant verification of mobile platform soundness With TPM or MTM, a "quote" command gives the possibility to prove to a distant entity platform soundness by signing a hash of one or several PCRs. Verification of platform soundness requires a perfect knowledge of these PCR values, depending of brand, type, versions, configurations, etc. Another approach is possible through the "sealed secure storage" capabilities of TPM/MTM. It is possible to define new keys which availability is conditioned by the fact that some PCRs after boot process have the same value than at sealing time, thus proving platform soundness. This give the possibility for a distant entity to verify trustworthiness of a platform without any knowledge of PCRs : this is also an example of TCG non-awareness.

2 Use case examples

In the TCG mobile working group, it has been considered as important to verify the adequacy of MTM specs regarding specific use cases, brought by different mobile operators or service providers. A document will be public soon. The chapters are :

SIMLock / Device Personalisation : Secure storage services provided by the MTM are the basis for this function which has to resist to attacks like deleting in the mobile the data linking mobile and one specific SIM.

Secure channel between UICC and mobile platform : As stated before, many sensitive mobile applications will need that some part of it is implemented in the SIM. But it would be totally unsecure if for example in the case of DRM or mobile TV the "content encryption key" computed inside the SIM was delivered in clear to the content decipherment function in the mobile platform. This is the reason why a specific topic has been defined around this issue, given that 3GPP and ETSI also address it.

Mobile Ticketing, Mobile Payment : In these applications, some counters like the ticket counter for a metro ticketing application, or the monetary balance of an electronic purse application are highly sensitive and require a high level of assurance for their protection. They will be implemented inside the SIM. But it remains for example the problem of "what you see is what you pay" ie integrity of user interaction which is totally related to the OS integrity.

DRM Implementation : Depending of the way it is implemented (right control and key computation inside or outside the SIM) , requirements on platform security are different, and quite obvious!

User Data Protection and Privacy : More and more applications are used on mobile phones. They are used to make phone calls, to send text messages, photos or short movies, to visit web sites, to send emails, to watch TV, to buy items (mobile payments), to take the bus (mobile ticketing), to access to the intranet of their companies, etc. The mobile is becoming a central tool in the life of its owner. However, if all the actions of the owner (user) can be linked together, it would lead to a major threat to the user's privacy. This trend is amplified by the introduction of NFC technology and by the Internet browsing capabilities which are now available on many mobiles. These evolutions give

mobile phones the capability to be used for many transactions in the day to day life, proximity (through NFC technology) or distant (through mobile network technology such as GSM/UMTS). Hence there is a strong need to provide the user with the guarantee that its privacy is protected by preventing third parties to link together the different actions of the user or to link them to the identity of the user. However, there is also a need to allow some kind of traceability of the user to counter misuses of a service and to keep the possibility to revoke mobiles in case of attacks resulting in the leakage of cryptographic material (keys). Hence although user privacy is important we often see that complete user anonymity is not desirable.

The protection of the user data is also important. The memory storage size of the mobile is steadily increasing and consequently, the users can and will store more and more important data (address book, text messages, emails, photos, payment receipt, electronic tickets, company's documents...) on their handset. It is important to guarantee that these data will not be stolen if the mobile is stolen or if a malicious application is run on the mobile. Data should also be protected against unauthorized modifications of such applications.

3 Privacy : overview of privacy enhancement technologies

More and more the "big brother" problem arises in the NTIC society, and is considered by consumers as a potential danger. This POC demonstrates that TPD can be used not only for usual authentication and key management issues, but can also help to enforce privacy, enabling for example anonymous secure access feature.

The terms "privacy" and "anonymity" may be distinguished, because anonymity signifies that a person can be entitled to some action, while remaining completely unknown (e-cash is an example), whereas privacy means that a person may well be known after an authentication, but it is guaranteed that no unauthorized entities are informed or that no data are generated which allow tracking of the person's activities. Such a possibility of tracking is also called traceability of person's activity.

Privacy as defined previously is related to regulations and organizational means to enforce these regulations : for example for an e-commerce company, to have computer customer databases which are protected, through access control, or to employ only "trustable" employees, etc.

As this paper doesn't deal with regulation and organization, we will address here only the PET technologies (privacy enhancement technologies). The two primary actors in PET are the prover and the verifier i.e. the entity which for example will deliver a service if it accepts the proof given by the prover.

Considering privacy, some improvements of classical identification/authentication protocols may help to improve it, and reduce the risk of misuse of user identity by the verifier, or an attacker between the two. Considering anonymity, i.e. technical methods for proving that you have some right without revealing who you are, we introduce here some known solutions for anonymity.

Nevertheless, these 2 concepts of privacy and anonymity are strongly related, and it is somehow difficult to establish a clear border between the two.

3.1 Anonymity with Pseudonyms

The mechanism which most of today's web sites (e.g. on-line auctions, on-line games, forums, etc) enable to ensure the anonymity of the users is based on the use of a pseudonym and a

password. This approach comes with a number of limitations. The probably most critical of them is that it does not provide a very convincing proof of authentication, since password is considered generally as a weak method : it may be stolen, used by someone else, etc. To prove to some entity that you have an access right to this entity is in some cases a very serious issue, where stronger mechanisms than a simple pseudo/password are to be used. Accountability requires also stronger mechanisms ; accountability is a property that becomes essential as high valuable transactions are at stake, like financial transactions, or even for example commitment for reserving a place in a hotel or in a flight.

3.2 Anonymity with attestation keys

Let us call a platform an entity owned by the prover (a person or an organization) and having some computational capabilities, like smart cards, mobiles phones or PDA, PC, TCU in a vehicle, etc. Public key cryptography is a mature technique, where certification authorities (CA) play an important role. Use of public key authentication or signature is a mean to obtain strength and accountability, as described before. The role of a Classic-CA is to attest to the world that a given prover has been assigned a certain public key pair. Clearly if only one public key pair is used by one prover, there is a full traceability of the actions of this prover. So a possibility is to use numerous public-keys pairs in order to improve privacy, and allow for frequent renewal of theses keys if necessary. And to rely on an infrastructure for obtaining certificates of these keys which guaranties that no association of these certificates to a unique prover identifier is possible.

This approach is taken for example by TCG : these numerous key pairs are called AIK (Attestation Identity Key). The overall role of the Privacy-CA is to vouch to the external world that a given platform is truly a trusted platform as defined by the TCG, and issue an AIK-Credential to state that fact. Thus, although privacy-CA uses the same mechanisms as a classic-CA to achieve similar aims, there are some underlying differences between the two. The Privacy-CA is a classic-CA that is also trusted not to disclose unique public keys attached to the platform, of registrants using AIK registration protocols. The privacy-CA respects then the notion of privacy or "blinding" (of a platform's true identity). As a summary of this approach, this method is a mean to obtain some anonymity of the prover, provided that privacy-CA follows their commitment of no disclosure. Under the same assumptions, untraceability is true for different AIKs, but traceability is possible for the usages based on the same AIK

Note : this approach enables anonymous attestation, but not "direct anonymous attestation" like described in 5, since it requires a privacy-CA on top of the prover and the verifier.

3.3 Anonymity with anonymous certificates

This technique has been chosen for the INSPIRED project [5]. Instead of relying on normal certificates obtained through a "privacy-CA", one can use anonymous or blinded certificates. Blind signature is a well known technique. Blind signature has been invented by D. Chaum, in the 80s[2]. It is possible to use this technique for obtaining certificates, since a certificate is

[2] Underlying mathematics are very simple with RSA, because of the multiplicative properties of RSA : if SIG is the exponentiation $x^d \bmod n$ function, where d is the secret exponent and n the public modulus, then $SIG(a.b) = SIG(a).SIG(b)$. "Blind message M" means choosing a random x, and computing $B = M.PUB(x)$, where PUB is the inverse of SIG function. So, having obtained a signature $SIG(B)$, unblind means dividing $SIG(B)$ by x, since $SIG(B) = SIG(M).x$

no more than a signature of a public key and some other data by a CA. In fact as the CA has to be aware of at least some of the data it signs (for example validity date, level of rights given by such a certificate...) the cryptographic technique which has to be used is "partial blind signature" instead of the simple "blind signature" technique. One example of such a protocol is given in annex 2.

This technique is used through two basic functions :

JOIN the platform obtains in a blinded way an anonymous certificate.

SIGN uses this anonymous certificate for a signature operation, without revealing any information which could, even with help of the Authority, reveal platform identity given in the JOIN phase. But the certificate is revealed, thus giving some traceability possibility.

After SIGN is performed, the verifier knows that the platform has been certified by such Authority through a JOIN protocol, but he doesn't know who the platform is.

Two modes of use are possible :

1. Attest that a JOIN with such authority has been done, by signing a challenge with SIGN.
2. Create a temporary key pair (ex : AIK in TCG), sign the public part with SIGN, and use this key pair for subsequent operations.

As a summary of this approach, we have a mean to obtain anonymity of the user without any assumption on the CA. Traceability is possible for the usages based on the same anonymous certificates, but combined with the previous technique, the same level of untraceability may be reached

Note : the word "group signature" is often used, with this kind of techniques : With SIGN, the prover (user) proves that he is part of the group of user who has obtained an anonymous certificate from such authority, without revealing who he is.

3.4 Anonymity with zero-knowledge proof of knowledge

This technique is more recent than previous one (2000 : see [6]) but more complex. It is a complex mix between blinding techniques and zero-knowledge. A short description of this protocol is given in annex. It requires also a large computational power that makes it more suited to PC than smart cards. This technique is used by TCG for a second form of anonymity, more powerful than the AIK technique, i.e., not needing assumptions on CA. For this reason, it is called DAA (direct anonymous attestation). It is used through two basic functions :

JOIN the platform obtains in a blinded way a "kind of certificate".

SIGN uses this "kind of certificate" for a "kind of signature" operation, without revealing any information which could, even with help of the Authority, reveal prover identity given in the JOIN phase. Particularly the "kind of certificate" is not revealed, thus preventing any traceability.

After the SIGN is performed, Verifier knows that the Platform has been certified by such Authority through a JOIN protocol, but he doesn't know who is the Platform and its owner.

Two modes of use of DAA are described in TCG :

1. Attest that a JOIN with such authority has been done, by signing a challenge with SIGN
2. Create a temporary key pair (ex : AIK in TCG), sign the public part with SIGN, and use this key pair for subsequent operations. So here, TCG gives a second possibility to obtain a AIK certificate, without the privacy-CA concept.

As a summary of this approach, it gives a mean to obtain anonymity of the user without any assumption on the CA. Traceability is also impossible

To some extent, it can be said that this technique is more powerful than the previous one. In practice, full untraceability is something dangerous, because there is no more any mean to "blacklist" a deviant user, ie a user having a bad behavior. Or it is impossible to blacklist a platform which has been stolen, or which has been compromised and its keys revealed. This is the reason why TCG has introduced some limitations to this technique[3], thus giving characteristics approximately equivalent to the previous one (cf. 3.3), at a higher price. This technique is used particularly by IBM in the IDEMIX project [7] and related products.

3.5 Implementation

In a TCG-aware environment, the first way is to use the DAA (Direct Anonymous Attestation) feature, as in PC platforms, to protect the privacy of the user during MTM attestation. The DAA allows proving that an AIK (Attestation Identity Key) was generated by a genuine TPM/MTM without revealing the corresponding EK (Endorsement Key) or using a privacy certification authority. The DAA signature mechanisms could also be used to give a proof that a piece of data is present in the mobile platform, or that a process (like verifying that the age of the user is > 18) has been performed in the trusted platform. However, the DAA mechanism is optional in the MTM specifications and so it may not be available in all mobiles.

In a TCG non-aware approach, we rely only on the integrity functions of the OS + secure storage capabilities enabled by MTM. Specific cryptographic protocols based on blind or group signature may be used to achieve privacy. For instance it is possible to prove to the outside worlds properties like : "I am a member of the group of users that have a genuine trusted mobile containing such piece of data (meaning for example that I have subscribed to such service) without revealing unnecessary information like my identity". These protocols can be run by specific applications devoted to access control & privacy, and the correct work of these applications will be guaranteed by the MTM using the secure boot feature for proving integrity of the platform & OS, and transitively of applications and secure storage of their secret keys. In the implementation described in 4, the crypto part of the process is implemented in the SIM, and user interactions are implemented with MIDP. So we are more in a "TCG unaware" approach.

Remark What is said in this paper would be a non sense if the low layer protocols for NFC (since we consider mainly proximity transactions) were designed in such a way that they need some kind of constant identifier, thus enabling to trace all the transactions coming from the same mobile. Fortunately, the ETSI standard TS 102 622 has defined a mode where this id (necessary for collision issues in contactless transmission) may be chosen randomly at each transaction.

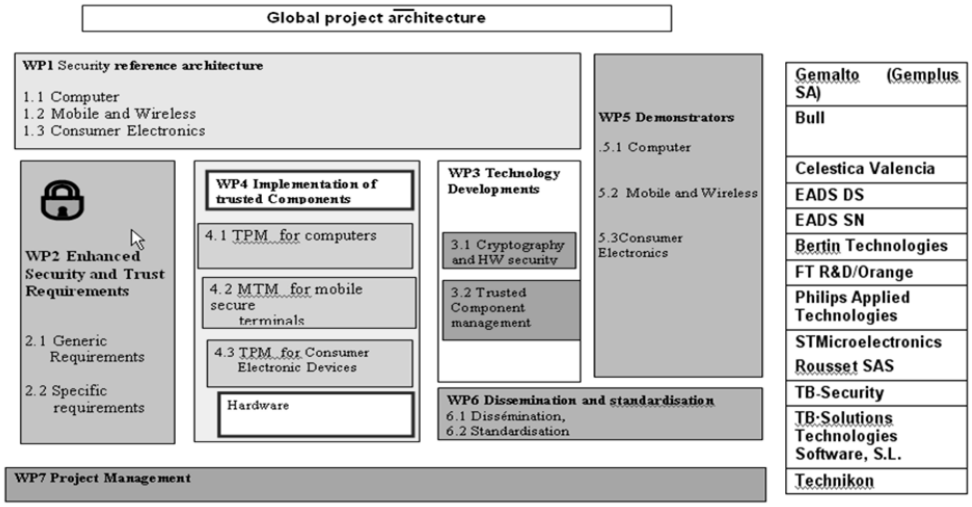
4 The TSC project (MEDEA)

4.1 Presentation

The Trusted and Secured Computing (TSC) project aims at developing a family of HW/embedded SW silicon components enforcing secure and trusted computing in the Consumer, Computer, Telecommunications and Wireless areas.

[3] For specialists : the role of the " gamma " variable in the DAA JOIN and SIGN protocols

It also intends to develop trust concept and architecture elements usable in other European industrial segments such as automotive, industrial, aeronautics (especially in their content acquisition and payment, ticketing and DRM aspects) Finally the TSC project will develop relevant European contributions related to Trusted Computing standards while keeping interoperability with existing US-led or Asian initiatives. The TSC partnership and organization is shown on the figure. Contribution described in 4 is part of WP5



4.2 Our demonstrator

Orange being an important mobile operator, our contribution to TSC is strongly linked to mobile services, considering particularly the emergence of NFC in the mobiles and the development of proximity transactions. So we have targeted the development of a general approach and architecture for control access to services taking care of privacy. The following drawing illustrates the architecture chosen for the MAACS project : Mobile Anonymous Access Control to Services. Let's take one example : for going to cinema and watch such film, and have a good rate, for example I have to prove that I am a student, and 18 years old at least. Usual solutions is to exhibit a student card, and an identity card for my age, thus giving far too much information on my student life, my age, my family, etc.

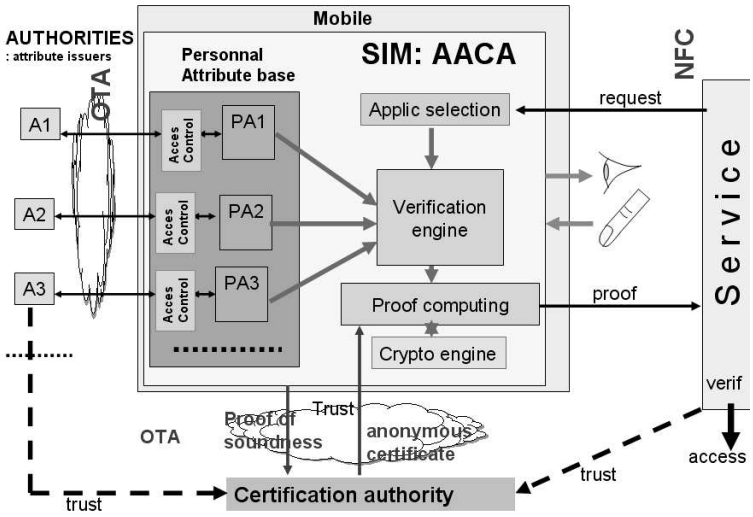
In our approach we store in the mobile (in fact in the SIM) different informations provided by different authorities (A1, A2..) which provision OTA a personal attribute base inside the SIM. This provisioning is of course secured through access control provided for example by GP (global platform) standard. The implementation of this Anonymous Access Control Application "AACA" has to be trusted, and so will get from a certification authority CA a certificate which will be anonymous through usage of one of the techniques described before. Finally, attributes authorities trust the CA, services trust the CA which trust the AACA. So, services trust the proof of attributes possession given by the AACA. Now, for example,

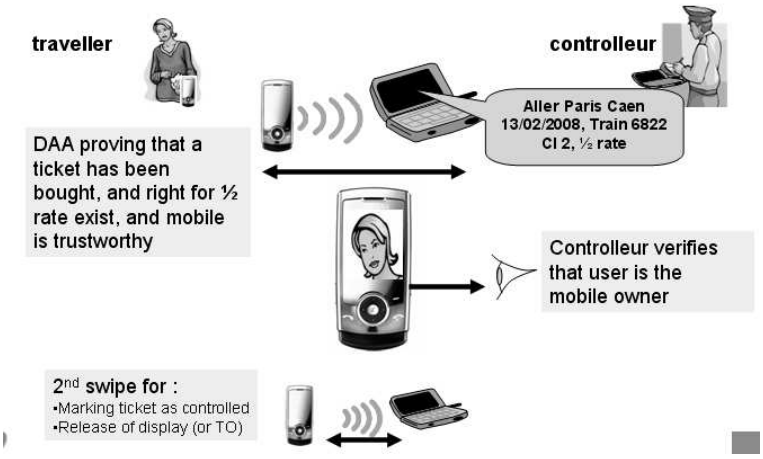
entering the cinema will simply results in the reception of a request from the terminal in the cinema, selection of the AACA application, verification that parameters "student" and "age >=18" fit with what is stored in the personal attribute base, and thus compute a proof using anonymous certificate, and send it to the cinema terminal. In some cases, it is possible that some acknowledgement is asked to the user through UI, and in this case, transaction need two "swipe" for receiving the request, and after, for sending the proof. This first example has no strong requirements for mobile platform trustability, since we have stated that all the process is done by the SIM. However, it could be argued that user interactions must not be attacked by some "malware", and that this case needs also trusted computing. .

The second example here is much more demanding in term of trusted platform.

Here, we consider the case of a personal right, which has to be used only by a given entitled person. This is the case for example for a train ticket with a half rate reduction. We could of course say that the photography of the person is sent to the controller terminal, but in this case, it is no more possible to speak about privacy !

So, this case requires that the photography of the traveller appears on its own mobile display, and controller verifies it. This verification has of course no value if the traveller mobile doesn't prove that it is trustworthy, since it is needed here to have an exclusive access to the mobile display between the two swipes. This property relies on a specific function which has to be embedded in the OS, and kept integer.





5 Conclusion

This walk through has addressed different aspects of trusted computing for mobiles : the progress of the work being done in TCG, some implementation impact with the discussion on TCG awareness, the issues around use cases and privacy, and finally very consumer oriented aspects with the example regarding our day to day transactions in the trains. We hope that this paper shows the importance of trusted computing, even in consumer real life use cases like described before, and also the problems around privacy, ie the difficulty to conciliate anonymity with the needs of identification and revocation in case of some attacks.

Références

1. TCG - Trusted computing platforms, trusting computing group - `https://www.trustedcomputinggroup.org/home`
2. Trusted Computing Group, TCG Mobile Trusted Module, Specification Technical Overview FAQ `https://www.trustedcomputinggroup.org/groups/mobile`.
3. Trusted Computing Group, Mobile Phone Work Group Use Case Scenarios, Specification Version 2.7, 2005 ; `https://www.trustedcomputinggroup.org/groups/mobile`.
4. ISO/IEC 9798-5 Standard. Part 5 : Mechanisms using zero-knowledge techniques - Second edition, December 2004.
5. Anonymous Attestation Using Blind Signatures : Inspired deliverable, 22/04/2005
6. Direct Anonymous Attestation : Ernie Brickell, Jan Camenish, Liqun Chen
7. IDEMIX : see the site of IBM-Zürich : `http://www.zurich.ibm.com/security/idemix`

Terms and Abbreviations

This document uses the following terms and abbreviations.

Term	Description
AIK	Attestation Identity Key
CA	Certification Authority
DRM	Digital Rights Management
EMV	Europay Mastercard Visa
ETSI SPC	European Telecommunications Standards Institute – Smart Card Platform
GSM	Global System Mobile
HW	Hardware
IC card	Integrated Circuit card or smart card
ME	Mobile Equipment
MNO	Mobile Network Operator
MPWG	Mobile Phone Workgroup
NFC	Near Field Communication
OMA	Open Mobile Alliance `http://www.openmobilealliance.org`
OS	Operating System
OTA	Over the Air
PIN	Personal Identification Number
PKI POS	Public Key Infrastructure Point of Sale terminal
RIM	Reference Integrity Metric
RTM	Root of Trust for Measuring
RTR	Root of Trust for Reporting
RTS	Root of Trust for Storage
RTV	Root of Trust for Verification
RTVI	Root of Trust for Verification Identifier
RVAI	Root Verification Authority Identifier public key configured in the CRTV for the platform root security authority.
SIM	Subscriber Identification Module. In the context of this paper, SIM, UICC, USIM have & the same signification.
SoC	System on Chip : primary platform host execution engine
SW	Software
TCG	Trusted Computing Group `http://www.trustedcomputinggroup.org`
TPM	Trusted Platform Module
TSS	TCG Software Stack
UICC	UMTS Integrated Circuit Card
UMTS	Universal Mobile Telecommunications System
USIM	Universal SIM
3GPP	3^{rd} Generation Partnership Project `http://www.3gpp.org`

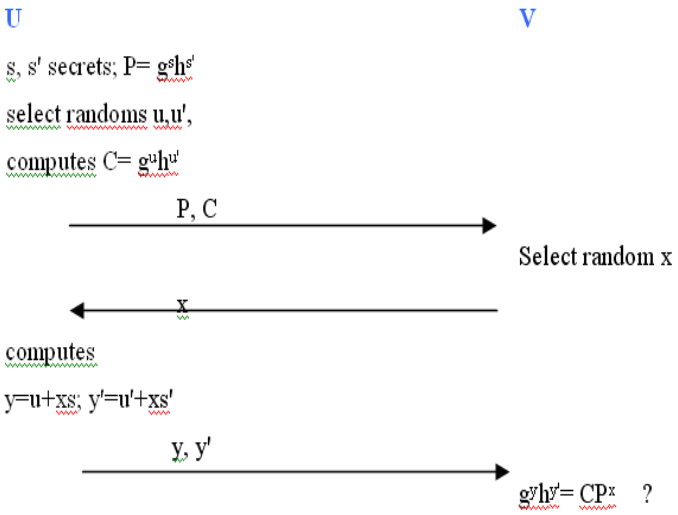
ANNEX 1 : the ZKPK protocol used in the TCG DAA function : overview

A full description of ZKPK can be found in [6], and the description of its implementation in the TPM as defined by TCG can be found on [1].

The discrete logarithm problem can be generalized :
- to the case of having several well chosen generators : for example having g and h , and c, and finding x and y such that $c = g^x h^y$ in Zn.
- where n is not a prime, but an RSA number, ie a product of two primes. In this case computing e-th roots in Zn is possible.

Starting from the SCHNORR or GPS protocols [4], ZKPK can be introduced as follow :

Proof that entity U knows s, s' such that $P = g^s h^{s'}$ The Schnorr protocol can be adapted as follows :



This kind of protocol is named a "proof of knowledge" of s and s' such that $P = g^s h^{s'}$, and notation used is : $\mathcal{PK}\ \{s, s'; P = g^s h^{s'}\}$. In this protocol, u and u' have only a circumstantial role, for blinding values s and s', which are the secrets of the commitment P, since this protocol can be seen as a commitment of U to prove to V that he knows these values s and s' such that $P = g^s h^{s'}$. Variables u and u' are often called "blinding variables" More over, this is P which is important, not C, which explains the notation above, where C doesn't appear. This mechanism can be applied to any number of commitments ; for example : $\mathcal{PK}\ \{s, z, r, r', r''; A = b^s g^z h^r, B = g^s h^{r'}, C = g^z h^{r''}\}$

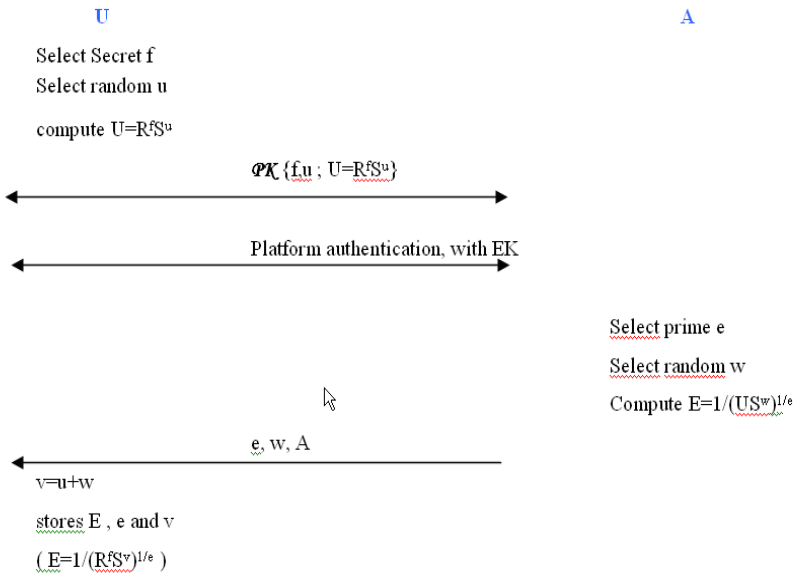
Proof that U knows a, b, d, and that $d = ab$ U has sent commitments $B = g^b h^{b'}, A = g^a h^{a'}$ and $D = g^d h^{d'}$; how to prove that $d = ab$? : D can be expressed by $D = B^a h^e$, with $b'a + e = d'$. So U can give : $\mathcal{PK}\ \{a, a', b, b', d, d', e; A = g^a h^{a'}, B = g^b h^{b'}, D = g^d h^{d'}, D = B^a h^e\}$ If $d = ab$, then U can perform this protocol. The difficulty of the discrete log problem can be used to prove that $d = ab$.

Proof that U knows v=c1/e , c public value, without revealing e, a secret of U Principle of the protocol : U "blinds" e by a random w, and will prove that he knows vg^w and w, such that $(vg^w)^e = cg^{ew}$; he must prove that the g exponent, z , is the product ew. For doing that he uses commitments C_w and C_z and prove that $C_z = C_w^e h^{r'''}$. So, U performs the

following protocol : U select random w, r, r', r'', r''' and computes : $z = ew$, $Cv = vg^w, C = cg^z h^r, C_w = g^w h^{r'}, C_z = g^z h^{r''}$; he sends Cv, and then performs :
$\mathcal{PK}$ $\{z, w, e, r, r', r'', r'''; (1)C/c = g^z h^r, (2)C_z = g^z h^{r''}, (3)C_w = g^w h^{r'}, (4)C = C_v^e h^r, (5)C_z = C_w^e h^{r'''}\}$
Clearly, if U knows e and v, he is able to perform this protocol. If U' is able to perform this protocol, then he knows $c^{1/e}$: Let $j = C_v/g^w$ Then $j^e = c$ since : $(2) + (3) + (5)$ imply that $z = ew$, since $(5) = g^{ew} h^{er''+r'''}$ and $(2) = g^z h^{r''}$ $(1) + (4)$ imply that $cg^z h^r = C_v^e h^r$, therefore that $C_v^e / g^{ew} = c$, therefore that $j^e = c$.
□ This protocol can't be used as a proof that U is certified by authority A which knows the factorization of n, and then can compute e-th roots over Zn. Indeed, there is no secret of U in the data which are certified by A. But it is the basis of TCG SIGN protocol.

Simplified TCG Protocol R, S, g, h are public elements of Zn, of great order In this protocol, user U has first to perform a JOIN procedure : he has to be identified and authenticated by authority A, using the TPM endorsement key EK. Authority A provides him with an e-th root of an expression $E = 1/R^f S^v$ where f is a secret chosen by U, and v is imposed by the authority (otherwise, it would be possible for U to choose f and u as multiples of e). In the SIGN procedure, U proves that he knows an e-th root of E, without revealing e nor u or v. So, User U proves that he is part of the group of users which have been certified by authority A, since only A is able to compute e-th roots.

JOIN

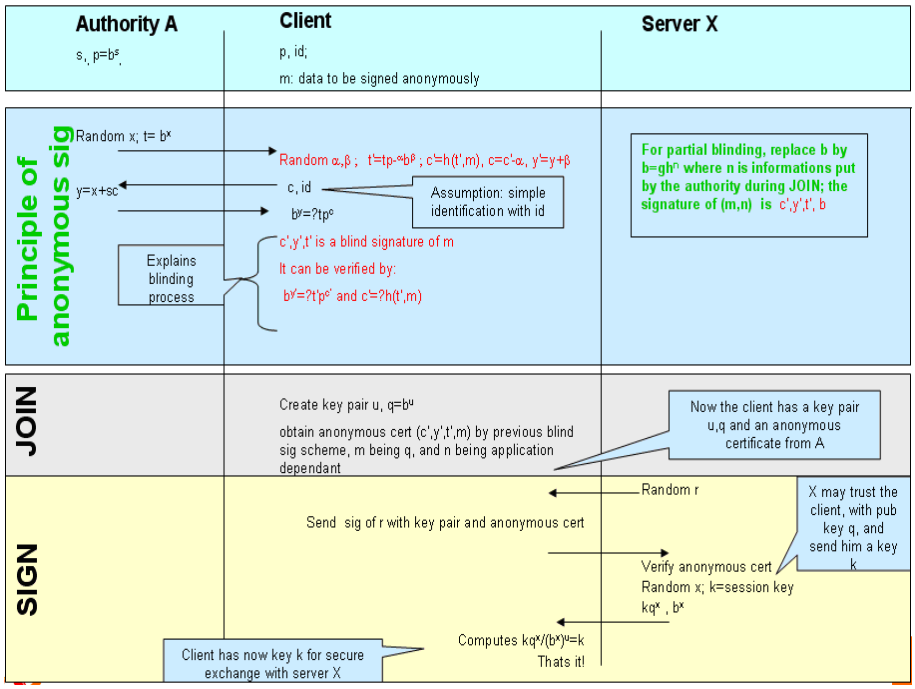


SIGN The SIGN procedure uses the technique of previous paragraphs. U performs the following protocol : U select random w, r, r_1, r_2 ; U computes : $T = Eh^w, z = ew, r_2 = r - er_1, C = g^z h^r = D^e h^{r_2}, D = gwh^{r_1}$ U sends T, and then performs : $\mathcal{PK}$ $\{w, r, r_1, r_2, e, z, f, v; C = g^z h^r, D = g^w h^{r_1}, C = D^e h^{r_2}, 1 = T^e R^f S^v h^{-z}\}$ The first three commitments prove that $z = ew$; the last one proves that U knows f, v and $(R^f S^v)^{1/e}$.

Conclusion This highly compacted description can be developed in term of elementary computations and exchanges. It shows the complexity of TCG-DAA protocol, since one has to take into account all the blinding variables of the proof of knowledge, and of the great number of computations required on U side. Indeed, U has to compute 20 exponentiations (10 for the commitment appearing in the SIGN protocol, multiplied by 2 for taking into account the blinding variables. Note that in TCG, not all the computations are performed by the TPM : part of the computations are done outside the TPM, in the PC, the goal being to limit the computation load in the TPM, without leaking any secrtes of the TPM, like f and v.

Annex 2 : partial blinding signature scheme

The protocol described here is patented. A Client gets an anonymous cert from a Certification authority A; Then he may use it to access securely and anonymously to a server X, and exchange with X a session key k; Two kind of information are considered : m and n. This information is chosen by the client, and part of the certificate. But m has to be hidden to the certification authority, whereas n has to be shown to it. One application is that m contains Client information, like name, address, etc, and n contains a version number, or a validity date, thus protecting a system against reuse of too old certificates. This possibility doesn't exist with classical Chaum blind signature scheme, where the whole data is hidden to the certification authority. The figure below summarizes this protocol, based on the Schnorr or GPS protocols : see [4]. In red : blinding operations are detailed. This protocol is much lighter than the previous one, and thus easier to implement on smart cards, and faster.



Digital Rights Management & Trust

Eric Diehl

THOMSON R&D France
1 Avenue Belle Fontaine
CESSON SEVIGNE, France
+33 2 99 27 32 54
eric.diehl@thomson.net

1 Introduction

Digital Rights Management (DRM) like any other content protection scheme is based on trust [1]. If trust is not possible, then the system will fail. A DRM designer should ensure that he builds his system on top of a sound realistic trust model. Furthermore, he should be able to trust the implementation of his system. He should also try to limit the risks by aligning the goals of every stakeholder. Each of the above recommendations can generate new interesting research topics.

2 Some research topics

2.1 Trust your model

Let us call our "old friends" : Alice and Bob. They often mutually trust each other. This is the case for OpenSSL. Unfortunately, this is not the right model when discussing DRM. In current DRM schemes, Alice does not trust Bob. New schemes of DRM are proposed where Alice trusts Bob [2]. Nevertheless, Alice may use techniques to detect unfaithful Bob. But even if Alice trusts Bob, this scheme may not be realistic. For instance, malicious Ruth may have installed a rootkit on Bob's computer. She now controls Bob's computer and may act illegally on behalf of Bob. This last trust model will become one of the most important ones in the coming years (and not only for DRM). This needs to thoroughly be explored.

2.2 Trust your implementation

Does the implementation of a DRM work as expected ? The first area to check is the effectiveness of trust model's implementation. For instance, many trust models (if not all) assume that some keys are kept secret. Too often, this basic assumption is wrong [3]. Even if the trust model is correctly implemented, DRM has to comply with robustness and compliance rules [4]. The robustness compliance is difficult to assess. Finally, there are the unavoidable flaws due to bad design or coding bugs. We need better tools to assess the security of software implementation, to validate the compliance to robustness rules and trust models.

2.3 Trust the greed for money

Many times, security failed due to bad incentives rather than bad designs [5]. When the entity that delivers security does not suffer the loss produced by a breach of this security, how can we trust this entity ? This is especially true in the case of DRM. The utilities, i.e. expected economic gains, of content providers are opposed to the utilities of consumers [6]. If Bob steals content from the DRM running on his platform, Alice will lose money, and not Bob (I do not speak with regards to potential litigations). The same if Bob illegally shares content. Game theory may allow finding new strategies or new equilibriums that would better align the incentives of each stakeholder. Game theory should be applied to the analysis of more complex business models. It should also be applied to define the optimal interaction between many players : content owners, content distributors, consumers and legal authorities. Finding a win-win position would be ideal.

3 Conclusion

DRM needs trust. We propose several tracks of research to build stronger trust. It means building more realistic (pragmatic ?) trust models. It requires better assessment of the security of implementations. DRM also needs also to better align the economic incentives of each player in order to reach a broader acceptance of DRM. All these would contribute to better trust, thus leading to more secure DRM.

Références

1. E. DIEHL, "A Four-Layer Model for Security of Digital Rights Management," Alexandria, Virginia, USA : ACM, 2008.
2. M. KIRSTEIN, Beyond Traditional DRM : Moving to Digital Watermarking & Fingerprinting in Media, MultiMedia Intelligence, 2008.
3. O. COURTAY and M. KARROUMI, "AACS under Fire," Security Newsletter, vol. 2, Jan. 2007 ; `http://eric-diehl.com/newsletterEn.html`.
4. Microsoft, "Windows Media DRM Compliance" ; `http://wmlicense.smdisp.net/wmdrmcompliance/`.
5. R. Anderson and T. Moore, "The economics of information security," Science (New York, N.Y.), vol. 314, Oct. 2006, pp. 610-3.
6. G.L. Heileman et al., "The drm game," Proceedings of the 2007 ACM workshop on Digital Rights Management, Alexandria, Virginia, USA : ACM, 2007, pp. 54-62 ; `http://portal.acm.org/citation.cfm?id=1314287`.

Vote électronique : constats, questions et certitudes

Chantal Enguehard

Chantal.Enguehard@lina.univ-nantes.fr
Université de Nantes
Laboratoire d'Informatique Nantes Atlantique
2, rue de la Houssinière
BP 92208
44322 Nantes Cedex 03
France
www.lina.univ-nantes.fr

Observatoire du Vote
ECCSI-NACCSI
Bastion Tower, 20
Place du Champ de Mars, 5
B 1060 Bruxelles
Belgique
www.observatoire-du-vote.net

Introduction

Le terme de "vote électronique" recouvre une mosaïque d'applications informatiques intervenant dans le cadre d'élections. Il peut s'agir du dépouillement de bulletins de vote digitalisés (e-counting), du recueil et du comptage des voix (ordinateurs de vote) ou encore de systèmes allant jusqu'à la gestion des émargements (vote par internet, vote par kiosques). Différentes typologies peuvent être envisagées selon le point de vue choisi pour structurer ce foisonnant domaine : vote en environnement surveillé ou vote à distance, centralisation ou décentralisation des procédures de comptage, matérialisation ou dématérialisation des bulletins de vote, c'est ce dernier fil conducteur que nous choisissons.

Dématérialisation du suffrage et des bulletins de vote

Les procédés de vote électronique dans lesquels bulletins et urnes sont dématérialisés comprennent les ordinateurs de vote dits DRE (Direct Recording Electronic), le vote par kiosque (vote à distance en environnement surveillé) et le vote par internet (vote à distance en environnement non surveillé).

Constats

Nous présentons les résultats de l'étude de l'Observatoire du Vote portant sur l'usage du vote électronique lors des quatre dernières élections politiques en France [1]. Cette étude mesure les différences entre le nombre de votes et le nombre d'émargements dans les bureaux de vote et constate que, dans l'échantillon étudié, ces différences sont plus importantes dans les bureaux de vote faisant usage d'ordinateurs de vote par rapport aux bureaux de vote pratiquant le vote à l'urne.

Questions

Plusieurs hypothèses sont explorées pour expliquer ces différences - nouveauté du procédé, agitation dans les bureaux de vote, taux de participation élevés - sans que l'une de ces hypothèses s'avère satisfaisante.

Certitudes

Nous démontrons que les processus informatiques ne peuvent être légitimement exclus du champ d'investigation sans pour autant qu'ils soient exploitables par voie de droit ni même, le cas échéant, en mesure de fournir les preuves de leur dysfonctionnement.

Bulletins rematérialisés

Nous examinons les systèmes de vote matérialisant les bulletins (Voter Verified Audit Trail) ou dépouillant des bulletins de vote matérialisés (e-counting). Nous démontrons que ceux-ci accroissent la complexité des procédures, ne pallient pas les défaillances des procédures de vote dématérialisé et introduisent des fragilités techniques, juridiques et organisationnelles.

Conclusion

Les systèmes de vote électronique n'apportent pas de plus-value déterminante quant à la confiance des électeurs dans le système électoral et sont perçus comme des systèmes favorisant la dissimulation de fraudes et incapables d'éviter des dysfonctionnements indétectables compte tenu de la nature anonyme des votes et de la dématérialisation des objets du vote. Ces faiblesses consubstantielles favorisent non seulement la défiance grandissante dans les systèmes électifs mais constituent aussi une menace de premier ordre pour la sécurité et la sureté nationale.

Références

1. C. Enguehard, Vote électronique - Élections présidentielle et législatives 2007 municipales et cantonales 2008. Observatoire du vote, 8 juillet 2008.

Secured and Practical Voting Machines

Emmanuel Bresson, Florent Chabaud, Xavier Chassagneux, Marion Videau

Direction centrale de la sécurité des systèmes d'information
51 boulevard de La Tour-Maubourg
75700 Paris cedex SP

Résumé Voting machines are used in France in political elections since 2004. Other examples, notably in the USA, have raised a lot of concerns and some criticisms against their use. The purpose of this paper is to propose new conception rules for voting machines, in order to improve their security to achieve the sincerity of the ballot, to obtain transparency of the voting and counting procedures, so as to raise the confidence of citizens. One way to reach this final goal, is to give to the citizens the ability to participate into control operations, which is easier to do in paper voting than in the case of voting machines.

Résumé Les machines à voter sont utilisées en France dans le cadre d'élections politiques depuis 2004. D'autres exemples d'utilisation, notamment aux USA, ont également donné lieu à un certain nombre de critiques, pour certaines justifiées, et alimenté la polémique sur leur utilisation. L'objet de cet article est de proposer des règles de conception nouvelles des machines à voter permettant d'améliorer la sécurité de ces machines pour garantir la sincérité du vote, d'assurer une transparence des opérations de vote et de dépouillement pour augmenter le niveau de confiance du citoyen dans ces équipements, de donner à tous les citoyens la possibilité de participer au contrôle des opérations de vote, rôle habituel dans le cas du vote papier, mais plus difficile à réaliser dans le cas du vote sur machine à voter.

1 Introduction

Voting machines are used in France in political elections since 2004. Their increasing use during some major ballots have caused number of questions on their robustness and the level of confidence one can have. For instance, one of the more detailed statistical analysis on the latest French elections have shown that 29.8% of the electronic voting offices have reported a difference between the number of signatures by hand on the register and the number of votes registered by the machine. Even if the difference may be non significant, this is far more than the 5.3% proportion of the traditional voting offices [1]. Other examples, notably in the USA, have raised a lot of concerns and some criticisms against their use.

We here focus on voting computers, where each elector must go to his/her usual election desk where he or she can register his/her vote on an electronic voting machine. As we will see, the proposals we make cannot apply on kiosks or internet voting systems. Our purpose is not to choose between the pros and the cons of electronic voting, but to propose new conception rules for voting computers, in order to :

– improve their security to guarantee the sincerity of the ballot,

– obtain transparency of the voting and counting procedures, so as to raise the confidence of citizens,
– give to the citizens the ability to participate into audit operations, which is traditional in paper voting but more difficult to achieve in the case of voting machines.

However, the other constraints we take in account are that voting :
– has to be technically feasible at a non overwhelming cost,
– must be easily usable for the vast majority of citizens !

2 Scientific foundations

Electronic voting have motivated a lot of research to achieve sincerity, confidentiality and auditability of the ballot. A lot of those researches have used cryptographic techniques, but few of them succeeded in designing practical voting schemes. One of the more interesting cryptographic result is the Scratch & Vote protocol [2]. But even if it is designed to minimize cost and complexity, the cryptographic skills that are necessary to understand the process, makes it hard to believe it is secure for a non cryptologist.

Other ideas have been proposed to make voting machines stick to the usual paper process. For instance, it is often suggested to print ballots and use a traditional counting procedure to verify the results announced by the machine. Those ideas suffer from technical potential problems : what if the printer is out of order ? But besides of that, this solution insists upon the fact that the electronic voting isn't reliable. What result would be considered correct if paper counting and automatic counting differs ? From a juridical point of view, such a question would be a mess... and actually such questions have led to a political mess in Florida in 2000, on the occasion of the US presidential elections.

We believe that a third approach is today possible [3,4]. It aims at separating the complex elaboration of voting interface from the more simple one of voting and counting. The latter, which is the core of the voting process for the citizens to be confident in, can then be published, and formally proved. In such a case, one can prove the sincerity of the voting process by advance.

Références

1. *Vote électronique - Elections présidentielles et législatives 2007 municipales et cantonales 2008* Chantal Enguehard, Observatoire du vote, 8 juillet 2008.
2. *Sratch & vote : self-contained paper-based cryptographic voting*, Ben Adida and Ronald L. Rivest, Proc. 5th ACM workshop on Privacy in electronic society, 29–40, ACM Press 2006.
3. *Prerendered User Interfaces for Higher-AssuVoting.* Ka-Ping Yee, David Wagner, Marti Hearst and Steven M. Bellovin, proc. EVT 2006. USENIX Press. `http://www.usenix.org/events/evt06/tech/full_papers/yee/yee_html/`
4. *Building Reliable Voting Machine Software* Ka-Ping Yee, PhD thesis, University of California, Berkeley, Fall 2007 `http://zesty.ca/pubs/yee-phd.pdf`.