La sécurité matérielle des systèmes embarqués

Lilian Bossuet, Guy Gogniat

1. Introduction

Les systèmes embarqués sont aujourd'hui de plus en plus présents dans de nombreuses applications. Le plus souvent sans fils et communicants, ils relèvent de flots de conception complexes qui jonglent avec des contraintes d'intégration et de fonctionnement (surface, consommation de puissance, débit,…) souvent très serrées. Depuis les années 2000, une nouvelle contrainte critique a émergé : la sécurité.

D'un point de vue application, les évolutions du commerce électronique sont limitées par la peur des utilisateurs de diffuser des données bancaires (ou privées) à travers un système et un canal de communications peu sûr. Cependant cela ne freine pas l'apparition de moyens de paiement dans les futures générations de téléphones portables. Ceux-ci doivent proposer des services de sécurité afin de rassurer le consommateur, par exemple les Smartphones G500 et G900 de Toshiba [TOS 07] embarquent un lecteur d'empreintes digitales. Mais, il s'agit dans ce cas uniquement d'un système de reconnaissance d'utilisateurs qui ne garantie pas la sécurité complète du système.

D'un point de vue du système, il est souvent nécessaire de protéger les données mémorisées en interne (ou une partie de celles-ci) et de garantir que l'utilisateur garde le contrôle du système en permanence. Ceci n'est pas simple car la plupart des systèmes embarqués sont des systèmes matériels communicants qui supportent une informatique enfouie, ils sont donc fortement menacés par des attaques logicielles et matérielles.

De plus, les systèmes embarqués sont au cœur d'un marché économique très dynamique avec de nombreux acteurs et soumis à de fortes contraintes (temps de

mise sur le marché « *time to market* » court par exemple). La concurrence entre les divers acteurs de la chaine de développement et de fabrication des systèmes est parfois rude. Ainsi, la protection de la conception et de la propriété intellectuelle est actuellement une problématique très importante pour les industriels.

Il apparait que, sous divers axes, la problématique de sécurité des systèmes embarqués, de leur conception, des données qu'ils mémorisent et qu'ils échangent, est un problème actuel majeur qui ne peut être traité que sous son angle logiciel. Les systèmes embarqués sont avant tout des systèmes matériels porteurs d'une informatique enfouie.

Ainsi, la sécurité matérielle est devenue une nouvelle dimension de l'espace de conception des systèmes électroniques (embarqués ou non) au même titre que la surface ou la consommation de puissance et d'énergie [KOC 04]. Ce chapitre propose une exploration des problématiques de sécurité des systèmes embarqués et donne quelques solutions matérielles originales et intéressantes.

2. Systèmes embarqués et problématique de sécurité

Si la sécurité est une nouvelle contrainte à prendre en compte lors des premières étapes du développement d'un système embarqué, alors elle doit être considérée simultanément avec les contraintes actuelles (consommation, surface, …). Les choix de conception ne peuvent s'effectuer qu'à travers des compromis entre certaines performances et le niveau de sécurité requis par l'application. La sécurité a toujours un coût qu'il est nécessaire d'évaluer précisément afin de choisir le meilleur compromis sécurité-performance.

2.1. *Contraintes de conception des systèmes embarqués*

L'appellation *système embarqué* regroupe de nombreux systèmes clairement distincts, tant au niveau des applications qu'au niveau des performances requises. De ce fait il est difficile de définir des contraintes universelles. Néanmoins, il est tout de même possible de lister un certain nombre de contraintes que l'on doit couramment respecter dans le cadre de la conception de systèmes embaqués, Parmi lesquelles nous trouvons :

- Le coût est, dans la plupart des cas, la plus importante des contraintes à prendre en compte lors du développement. Il s'agit d'une contrainte complexe à évaluer étant donné le nombre de paramètres qui peuvent entrer en jeu lors de son évaluation. Cette contrainte est particulièrement forte dans le cas d'applications grand public (fort volume de production).

- La limitation des ressources de traitement est liée à une limitation de la surface disponible pour le système et pour les composants qui le constitue. Il s'agit aussi d'une limitation des ressources de mémorisation. Ces deux contraintes restreignent fortement la complexité des algorithmes que l'on peut implanter efficacement sur ces systèmes.

- Le débit de données en entrée et en sortie est aussi limité. Néanmoins, la tendance actuelle des applications va vers une augmentation des besoins en termes de vitesse et de quantité des échanges de données, donc vers une augmentation forte des débits. Il s'agit là d'un goulot d'étranglement pour la conception et d'une contrainte difficile à respecter du fait des précédentes contraintes.

- Le nombre de connexions du système vers l'extérieur est limité, physiquement (nombre d'entrées/sorties par composant) et aussi en performances (nous l'avons vu avec la limitation concernant les débits).

- La limitation en termes de consommation de puissance et d'énergie est souvent importante car un système embarqué n'est qu'une partie d'un système hôte qui peut être lui-même limité en source d'énergie. C'est bien entendu le cas de tout système alimenté par une batterie. Effectivement, la taille de la batterie (donc l'encombrement et le poids de celle-ci) est liée à la puissance, sa vitesse de charge et décharge est liée à l'énergie.

- Les besoins de flexibilité logicielle et matérielle sont souvent grands. D'une part la flexibilité permet une meilleure intégration du système dans un large domaine d'application. D'autre part elle permet au système d'évoluer dans le temps par des mises à jour logicielles et/ou matérielles grâce à l'utilisation de mémoires programmables et de circuits matériels reconfigurables.

Nous pourrions continuer d'énumérer la liste des contraintes usuelles, cependant l'espace de conception est déjà sérieusement réduit par cette première approche. Les solutions pour répondre à ces contraintes sont connues bien qu'en constante évolution. L'hétérogénéité des ressources, logicielles et matérielles, disponibles sur le système, l'utilisation de flots de conception conjoints, sont des exemples de solutions actuellement employées.

2.2. *La problématique de sécurité des systèmes embarqués*

De nombreux acteurs interviennent dans le développement, la fabrication, la mise œuvre, la mise au point et l'utilisation d'un système embarqué. Le nombre

d'acteurs dépend de la complexité de l'application. Le besoin en terme de sécurité (ou protection) des différentes entités ne sont pas les mêmes.

Considérons, par exemple, un système mobile pour les communications à distance et pour des applications multimédia (musique, vidéo à la demande,…). Nous pouvons, dans une première approche, considérer six acteurs distincts prenant part à la vie du système (du développement à l'utilisation) et donner leurs besoins en termes de sécurité et de protection.

- Les fabricants de composants logiciels (systèmes d'exploitation par exemple) ou matériels (composants virtuels numériques, circuits intégrés pour le traitement en bande de base, composants discrets pour les radiofréquences,…) ont besoin de protéger leurs propriétés intellectuelles. Dans un marché fortement concurrentiel l'espionnage industriel est un risque de pertes financières importantes. Les entreprises impliquées à ce niveau de la conception doivent se prémunir contre toutes attaques visant à copier illégalement un composant pour le revendre frauduleusement ou visant à étudier un composant afin de pouvoir rapidement proposer sur le marché un composant concurrent amélioré (on parle en anglais de *reverse engineering*).

- Les fabricants de systèmes embarqués (intégrateurs des composants logiciels et matériels) sont face aux mêmes problèmes de sécurité. Ils doivent se prémunir contre l'espionnage industriel.

- Les fournisseurs de services doivent se prémunir de toute utilisation frauduleuse de leurs services. Ils doivent donc sécuriser l'accès aux services et mettre en place des systèmes pour s'assurer qu'une personne utilisant leur service a bien l'autorisation de le faire.

- Les fournisseurs d'applications (par exemple des services de paiement à distance) doivent apporter aux utilisateurs les garanties nécessaires pour protéger leurs données. Ils doivent aussi s'assurer de l'authentification de l'utilisateur. Enfin, ils peuvent aussi se trouver face aux problématiques de protection de la propriété intellectuelle pour leurs applications.

- Les fournisseurs de contenus (musique, vidéo,…) ont besoin de gérer les droits numériques d'utilisations (DRM, *Digital Rights Management*). Il s'agit de s'assurer que l'utilisation des contenus est autorisée et d'éviter la copie non autorisée. Les techniques d'échanges de fichiers *peer-to-peer* ont mis en lumière ces problématiques de protection des droits d'utilisation des contenus numériques.

- Enfin, l'utilisateur final peut mémoriser dans son système de plus en plus d'informations personnelles, parfois même sensibles (cas de l'utilisation professionnelle d'un téléphone portable par exemple). Ces informations doivent être protégées contre le vol. Des systèmes comme les téléphones mobiles, embarqueront rapidement des applications de paiement électronique (avec l'intégration de puce RFID[1] par exemple). Pour de telles applications, le système doit assurer à l'utilisateur qu'il est le seul en mesure de pouvoir effectuer un paiement.

On le voit, les acteurs concernés par la sécurité des systèmes embarqués sont très nombreux. L'exemple que nous proposons est didactique, mais il peut facilement être transposé à d'autres exemples pour lesquels le nombre d'acteurs peut être plus ou moins important. Il est à noter que les différents acteurs peuvent très bien être victimes ou attaquants pour un même système. Les attaques mises en œuvre peuvent viser, par exemple, à détruire le système, le détourner de son fonctionnement normal, le prendre en main, le rendre inutilisable (on parle alors de dénie de service) ou encore lui extorquer des informations sensibles. Pour atteindre ces objectifs, l'attaquant utilise aussi bien des techniques logicielles que matérielles.

2.3. *Les principales menaces de sécurité*

Le développement d'un système sécurisé ne peut se faire sans un modèle de menaces. Ce modèle permet de déterminer les attaques auxquelles le système doit répondre ainsi que leurs coûts. Il est possible, par exemple, d'établir que le coût de mise en œuvre de certaines menaces est prohibitif par rapport à la valeur de la cible. Dans ce cas, l'attaque ne doit pas forcément être considérée comme une menace (sauf dans le cas où l'attaquant a beaucoup de moyens et ne se soucie pas du coût).

Les pires cas, c'est-à-dire les plus exposés aux menaces sont les systèmes mobiles communicants embarquant un système d'exploitation enfoui, comme le montre la figure 1. Effectivement, ces systèmes sont potentiellement sensibles aux attaques sur le canal de communication. Ils doivent donc assurer les services de confidentialité, d'authentification et de non répudiation des données qu'ils transmettent sur le canal de communications. Mais ils sont aussi sensibles aux attaques logicielles et matérielles.

---

[1] La technologie RFID (*Radio Frequency Identification*) permet des communications à très faible distance (15 cm), elle est souvent utilisée pour des applications de paiement sans contact, par exemple dans les transports en communs comme pour le tramway Bordelais.

Figure 1. *Menaces contre les systèmes embarqués.*

Le canal de communication peut être utilisé pour envoyer un programme malicieux qui conduit à une attaque dite logicielle. Le téléchargement de ce programme dans le système entraine l'exécution de codes plus au moins hostiles comme des vers, des virus, des chevaux de Troie ou des bombes logiques. Le lecteur intéressé pourra lire [FIL 06] pour une classification et une description de ces différentes attaques. Les systèmes embarqués, tout comme les ordinateurs de bureaux ou portables, sont donc des cibles potentielles pour les attaques logicielles dès lors qu'ils embarquent un système d'exploitation [DAG 04].

Ces systèmes, comme les cartes à puces, sont des systèmes matériels qui peuvent stocker des informations sensibles et utiliser des primitives de sécurité pour remplir les services nécessaires à la protection des échanges de données (confidentialité, intégrité, non-répudiation). Ces primitives sont, par exemple, des algorithmes de chiffrement asymétrique (à clé publique et à clé privée) ou symétrique (une seule clé secrète utilisée pour le chiffrement et le déchiffrement), des algorithmes de hachage qui produisent une marque utilisable pour garantir l'intégrité des données, un mélange des deux pour proposer par exemple les services de non-répudiation [MEN 96]. Ces primitives, une fois implantées dans le matériel, sont sensibles à des attaques matérielles par effraction ou sans effraction (les termes intrusives et non intrusives sont aussi utilisés en français, les termes *invasives* et *non-invasives* sont les équivalents anglais).

Les attaques matérielles par effraction portent généralement atteinte à l'intégrité du composant qui ne fonctionne plus correctement (voire plus du tout) une fois l'attaque effectuée. Lors de ces attaques sur un circuit intégré, l'attaquant extrait la puce du boitier. Une fois nue, la puce peut être découpée pour permettre la reconstruction du plan de câblage interne (*layout*) [AND 01]. Il s'agit alors d'un processus d'ingénierie inverse. Cette technique couramment utilisée dans les cas d'espionnage industriel est couteuse et demande beaucoup de matériel, ainsi que de grandes compétences techniques hétérogènes.

Certaines attaques matérielles sont dites par effraction sans toutefois détruire le circuit (semi-intrusive ou *semi-invasive*). C'est le cas des attaques par injection de fautes. Ces fautes peuvent être de diverses formes, glitchs (courtes transitions parasites) sur l'horloge ou sur la tension d'alimentation, injection de fautes par faisceaux lumineux intenses (laser par exemple) [SKO 02]. Les fautes peuvent être utilisées pour modifier le fonctionnement du circuit [CHO 05] ou pour propager une erreur le long du chemin de données (attaque dite différentielle ou DFA pour *Differential Fault Analysis*) [GIR 03]. Dans les deux cas il s'agit de forcer le circuit à sortir une information ne pouvant être délivrée en fonctionnement normal. Celle-ci peut permettre d'extraire une donnée sensible comme une clé de chiffrement. Les attaques par injection de fautes demandent, elles aussi, du matériel couteux et de fortes compétences techniques. La caractérisation précise de l'injection de faute, comme par exemple la localisation physique du point d'injection pour les attaques optiques, peut prendre beaucoup de temps et nécessite parfois une pré-attaque matérielle avec effraction et de l'ingénierie inverse.

Les attaques sans effraction sont beaucoup plus simples à mettre en œuvre et demandent moins de matériel et de compétences. On les appelle souvent attaques sur canaux cachés puisque le principe de ces attaques est d'analyser le comportement du circuit lors de son fonctionnement. L'analyse peut se faire sur la mesure de la consommation de puissance. S'il est possible d'effectuer une corrélation entre la mesure de la consommation de puissance et par exemple la clé utilisée dans un algorithme de chiffrement, alors il est possible de découvrir cette clé par analyse de cette mesure. C'est le cas lors de l'utilisation d'un algorithme de chiffrement symétrique (même clé secrète pour le chiffrement et le déchiffrement) comme l'AES [NBS 01] implanté dans un circuit matériel. Effectivement, la technologie des transistors MOS (*Metal Oxyde Semiconductor*) utilisée dans ce cas, a pour caractéristique une consommation dynamique de puissance qui dépend des commutations des transistors et donc des signaux internes. Cette caractéristique est utilisée très efficacement dans l'attaque DPA (*Differential Power Analysis*) développée en 1999 [KOC 99]. Cette attaque permet dans un temps très court (quelques minutes) et avec peu de matériel (un oscilloscope et un ordinateur) de découvrir la clé de chiffrement ou de déchiffrement codée sur 128 bits utilisée dans l'algorithme AES. Même si ce dernier est mathématiquement incassable avec les

moyens informatiques actuels. De même des attaques par analyse du rayonnement électromagnétique DEMA (*Differential Electromagnetic Analysis*) [QUI 01] ou par analyse temporelle des signaux en sorties [KOC 96], permettent aussi d'extraire des informations sensibles.

Les attaques matérielles sont nombreuses et variées. Leur efficacité, leurs coûts définissent un large espace d'attaques pouvant viser les systèmes embarqués. Des contremesures, ou protections, sont couramment développées afin de contrer ces attaques. Cependant la recherche autour des attaques avance rapidement et il est nécessaire de faire évoluer en permanence les mesures de protection. L'idée importante est qu'une fois réalisé matériellement, tout système devient sensible aux attaques matérielles qui peuvent être parfois d'une efficacité redoutable.

L'utilisation conjointe de techniques d'attaques logicielles et matérielles, sur le système et sur le canal de communication, peut gravement porter atteinte à la sécurité du système et des données qu'il échange et qu'il mémorise. La sécurisation du système et de ses données est donc un problème complexe pour lequel il faut rechercher des solutions dans un large espace de conception (logicielle et matérielle) que nous allons explorer dans la suite.

3. Sécurité des systèmes et des données

Un système embarqué n'est pas un ordinateur de bureau ni une carte à puce. C'est pourquoi il est nécessaire de proposer des solutions de sécurité qui prennent en compte les spécificités de ces systèmes, telles que les contraintes de développement présentées au chapitre 2.1. Néanmoins, la sécurité des systèmes logiciels dédiés (ordinateur de bureau) et des systèmes matériels sécurisés (carte à puce) est un domaine largement étudié. Il convient donc de s'en inspirer pour adapter les concepts aux systèmes embarqués.

3.1. Principe de sécurité en profondeur (projet ICTER)

Le projet ANR blanc ICTER [ICTER 06] a pour objectif d'analyser les potentialités, en termes de sécurité, des plateformes matérielles reconfigurables pour les systèmes embarqués. Les plateformes matérielles reconfigurables (tels que les circuits FPGA[2]) apportent un compromis entre la flexibilité des plateformes

[2] Les FPGA (*Field Programmable Gate Array*) sont des circuits intégrés numériques. Ils sont constitués d'un grand nombre d'éléments dont la fonction est configurable (éléments logiques, arithmétiques, de mémorisation, d'entrées-sorties) souvent agencés sous forme de matrice. Ces éléments sont reliés par un réseau très

logicielles (à base de microprocesseurs) grâce à la mise à jour de la structure par reconfiguration et les performances des circuits intégrés matériels spécifiques à une application (ASIC, *Application Specific Integrated Circuit*) grâce, par exemple, à une implémentation parallèle des algorithmes [BOS 04a].

Pour débuter cette étude, les acteurs du projet ICTER proposent d'étudier la sécurité d'un point de vue logiciel et d'un point de vue matériel. Concernant la vision matérielle, ils proposent de trouver des solutions aux différentes attaques depuis le niveau système jusqu'au niveau physique (technologique). La figure 2 schématise par une pyramide le concept de *sécurité en profondeur*.

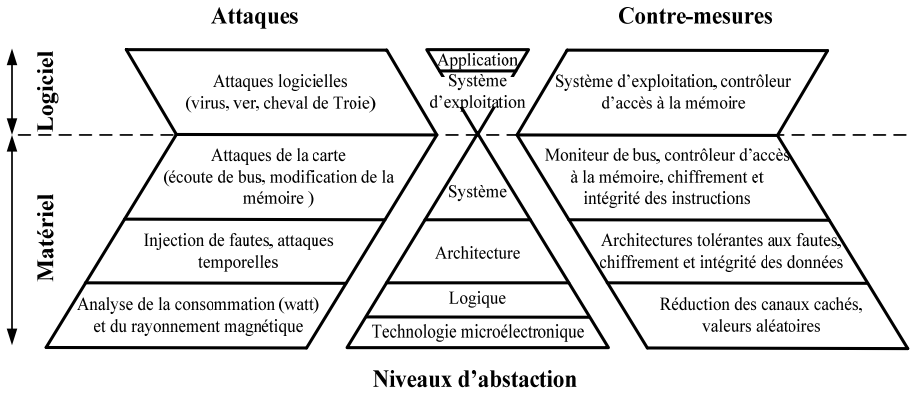

Figure 2. *Pyramide de sécurité : vers une défense en profondeur.*

La profondeur de la sécurité s'oppose à un traitement « en surface » de celle-ci. Effectivement, la vision proposée par le projet ICTER est de prendre en compte, à chaque niveau (logiciel et matériel) de la conception du système, la notion de sécurité. En effet, si le concepteur apporte une solution de sécurité à un niveau et qu'il néglige un niveau supérieur ou inférieur, il peut laisser ouverte une faille de sécurité qu'un attaquant pourra vite exploiter en utilisant une variété d'attaques logicielles et/ou matérielles.

Une analogie classique avec la conception de système à basse consommation de puissance (domaine bien connu pour les concepteurs de systèmes embarqués) est souvent faite. Car de la même façon, des optimisations faites, par exemple, à haut niveau peuvent être dégradées par une technologie peu optimisée et inversement [HAV 00].

---

dense d'interconnexions configurables. La configuration est mémorisée dans une mémoire SRAM ou Flash, ou avec des éléments fixes du type Antifusible [BOS 06].

Sur la gauche de la figure 2 quelques attaques bien connues peuvent nous permettre de dégager les caractéristiques que devraient avoir un système embarqué pour être sécurisé (sécurité du système et des données embarquées). C'est ce que nous allons détailler dans la suite.

3.2. Caractéristiques d'un système matériel embarqué sécurisé

Un système embarqué sécurisé doit avoir certaines caractéristiques. Il n'est pas nécessaire que toutes ces caractéristiques soient réunies pour un même système, cela dépend du modèle de menaces que l'on souhaite adresser. Afin de renforcer la sécurité d'un système au niveau matériel et ainsi empêcher (ou rendre plus complexes) les attaques, les points suivants doivent être considérés (les mots en italiques indiquent les terminologies anglaises utilisées dans la littérature) [GOG 06]:

- Le système doit être continuellement conscient de son état et notamment de ses faiblesses afin de réagir si nécessaire. Le système doit être **conscient de sa sécurité** (*Security Aware*).

- Le système doit analyser son état et celui de son environnement afin de détecter toutes activités anormales. Il doit embarquer des capteurs et des moniteurs afin de contrôler son activité. Le système doit être **conscient de son activité** (*Activity Aware*).

- Le système doit être capable de réagir rapidement vis-à-vis d'une attaque ou d'anticiper les attaques. Le système doit être **agile** (*Agile System*).

- Le système doit être capable de mettre à jour ses mécanismes (logiciels et matériels) de sécurité en fonction des évolutions des attaques. Le système doit être **évolutif** (*Software and Hardware Update*).

- Le système ne doit pas laisser échapper d'informations (fuites de données) afin de ne pas permettre les attaques passives. Le système doit être **sans symptôme** (*Symptom Free*).

- Le système doit être capable de supporter des attaques physiques. Le système doit être **résistant** (*Tamper Resistant*).

Parallèlement, le système doit fournir des performances élevées afin de respecter les contraintes des applications. Débit, latence, surface, puissance, énergie sont autant de paramètres devant être adressés simultanément afin d'intégrer efficacement les applications actuelles et à venir. Il est donc nécessaire de développer des flots de

conception qui prennent en compte la sécurité comme une contrainte supplémentaire tout en respectant les autres contraintes [SCH 03].

3.3. Solutions matérielles de sécurité, du niveau système au niveau physique

Un système embarqué est un système complexe et souvent fortement hétérogène. On y trouve classiquement des parties programmables (microprocesseurs), des systèmes de communications internes (bus, réseau sur puce), de la mémoire (instructions, données, configurations), des unités de contrôles, des entrées-sorties, des parties matérielles (reconfigurables ou non), divers périphériques (communications vers l'extérieur par exemple). Toutes ces entités peuvent être astucieusement détournées de leur fonction pendant une attaque, mais aussi être utilisées au profit de la protection du système et des données. Le but est alors de donner au système les caractéristiques vues précédemment.

3.3.1. *Solutions matérielles de sécurité au niveau système*

A ce niveau, le système est considéré dans sa globalité. C'est à ce niveau qu'il est possible d'analyser en permanence l'activité interne et externe du système afin de détecter toutes opérations irrégulières. L'activité externe est mesurée par des capteurs (température, tension d'alimentation …) et l'analyse confronte les mesures en fonctionnement avec des caractéristiques de fonctionnement normal [WOL 06]. Le système doit embarquer un contrôleur dont le rôle est de détecter automatiquement une activité externe suspicieuse. L'activité interne est contrôlée par des moniteurs placés sur le(s) réseau(x) de communication(s) interne(s). Les échanges de données entre les différents composants de l'architecture interne du système sont surveillés et comparés à une activité en fonctionnement normal [ARO 05].

Que ce soit pour une surveillance interne ou externe, il est nécessaire de pouvoir connaitre précisément le comportement du système dans le cadre d'un fonctionnement normal. Ce qui n'est pas toujours simple, puisqu'un fonctionnement normal peut être caractérisé par un spectre large. S'il est trop large, il y a un risque qu'un fonctionnement altéré dû à une attaque ne soit pas détecté. S'il est très réduit, il y a cette fois un risque de faire des détections de fausses attaques, l'analyse pouvant être dupée par un changement normal de l'environnement (augmentation de la température par exemple). Dans tous les cas, le fonctionnement du système doit être déterministe (ce qui ne veut pas dire que certains calculs ne soient pas de type aléatoires).

Une fois une activité *anormale* détectée le système doit réagir. Pour cela il doit être agile. Par exemple il peut passer d'une configuration en mode de

fonctionnement normal à une configuration en mode sécurisé. Dans ce mode, certaines données peuvent être détruire (car parfois il est préférable de perdre des données que de les révéler), certaines fonctions peuvent être bloquées (comme les communications vers l'extérieur) et une expertise de l'utilisateur peut être sollicitée. Pour mettre ces services en place, le système doit être reconfigurable. Cela peut se faire par le biais d'un système programmable ou avec du matériel reconfigurable. Cette dernière solution, bien que pouvant être complexe dans sa mise en œuvre, est efficace car elle permet d'avoir un système performant en fonctionnement normal [GOG 06].

Dans tout les cas il est nécessaire de s'appuyer sur un réseau de communication interne sûr. Par exemple, il peut être efficace de chiffrer les données échangées sur le bus. Cependant cette solution peut dégrader fortement les performances temporelles (fréquence de fonctionnement, débit en entrée-sortie) du système. C'est pourquoi il est très intéressant de considérer la sécurité des données échangées dès la conception du réseau de communication. Par exemple, [EVA 05] propose une solution intéressante de réseaux sur puce intégrant un système de sécurité des données.

3.3.2. *Solutions matérielles de sécurité au niveau architecture*

A ce niveau un seul module est considéré (microprocesseur, accélérateur matériel, mémoire …). L'architecture de ces modules doit être flexible, efficace et fiable (tolérance aux fautes) sans fournir des informations sur des canaux cachés.

De nombreuses études ont été réalisées sur l'implémentation efficace d'algorithmes assurant les services de confidentialité, intégrité et non-répudiation des données (algorithmes de chiffrement asymétrique, de chiffrement symétrique et de hachage). Effectivement, la distance entre l'expression mathématique d'un algorithme de chiffrement et sa réalisation logicielle et/ou matérielle peut être grande. Par exemple lors du développement du standard de chiffrement symétrique AES [NBS 01] une compétition internationale fut ouverte afin de proposer la meilleure solution d'implémentation logicielle et matérielle. Ce sont des chercheurs belges, J. Daemen and V. Rijmen, qui ont remporté la compétition grâce à l'efficacité de leur architecture pour les implémentations matérielles. L'architecture standardisée de l'algorithme de chiffrement AES porte aujourd'hui le nom de Rijndael en référence aux deux vainqueurs [DAE 02].

Des cibles sont particulièrement prisées pour l'implémentation matérielle des algorithmes de sécurité des données. Ce sont les circuits matériels reconfigurables FPGA. Effectivement, l'architecture de ces circuits utilise un nombre très important (plusieurs dizaines de milliers) d'éléments logiques de grain fin (pouvant réaliser des fonctions logiques à quatre entrées sur un bit). Cette granularité est très bien

adaptée aux calculs utilisés dans de nombreux algorithmes de chiffrement [WOL 04]. De plus, ces composants, lorsqu'ils sont de technologie SRAM ou FLASH (environ 90% du marché) sont reconfigurables. Cette propriété implique la possibilité de modifier l'algorithme implanté dans le circuit par reconfiguration matérielle. Cela permet au système de pouvoir évoluer dans le temps par une mise à jour matérielle. Ainsi, un nouvel algorithme peut en remplacer un autre, ou une nouvelle architecture intégrant des contremesures à une attaque peut être implantée.

Comme nous l'avons vue précédemment il est possible d'attaquer une implantation matérielle d'un algorithme de chiffrement (pour trouver la clé de chiffrement ou de déchiffrement) par injection de fautes. Ces techniques d'injection de fautes sont principalement issues des travaux sur le test et l'évaluation de la fiabilité des composants intégrés. Ces travaux ont abouti au développement de techniques permettant au composant de résister à ces fautes, on parle alors de tolérance aux fautes. Ces mêmes techniques peuvent être mises en œuvre pour sécuriser un système matériel contre les attaques par injection de fautes. Par exemple, des architectures concurrentes peuvent exécuter le même calcul en parallèle, de sorte que si une des architectures est attaquée le résultat puisse être validé sur une autre architecture par un vote majoritaire [KAR 02]. Bien sûr, ce genre de solution conduit à un surcoût très important en surface de silicium et en consommation de puissance. Les techniques de détection et correction d'erreurs (aussi utilisées en télécommunication pour se prémunir des erreurs produites par un canal bruité) peuvent être utilisées avec un surcoût réduit [BER 03].

Au niveau architectural, il est possible de réduire, voir d'annuler, certaines informations sur des canaux cachés qui sont exploités lors des attaques sans effraction (voir section 2.3). Par exemple, la consommation de puissance est utilisée dans les attaques de type DPA pour retrouver la clé de chiffrement d'un algorithme de chiffrement symétrique. Un dispositif interne peut brouiller cette information (ajout de bruit de consommation) ou la lisser comme dans [MES 05]. L'attaque DPA, comme nous l'avons expliqué, se fait grâce à une corrélation entre la clé de chiffrement utilisée lors de la génération du texte chiffré et la consommation de puissance du circuit durant cette opération. Il est possible d'éliminer cette corrélation en ajoutant un masque (nombre aléatoire) dans le flot de calcul [MES 01]. Cet ajout n'a aucun effet sur le résultat ni sur la complexité des calculs.

Les générateurs de nombres aléatoires vrais sont des circuits très couramment utilisés pour les systèmes sécurisés ou de sécurité. Ils sont utilisés pour générer des masques pour des contremesures, mais surtout ils sont utilisés pour générer des clés utilisées par les algorithmes de chiffrement. La problématique pour ces systèmes est de satisfaire des tests drastiques prouvant de façon probabiliste qu'ils génèrent bien un flot de bits de façon aléatoire [NEC 01]. Toute génération pseudo-aléatoire entrainerait un risque important de reconstitution des clés. Pour satisfaire les

exigences aléatoires, ces générateurs utilisent le plus souvent des phénomènes aléatoires physiques qui sont considérés comme des bruits *gênants* par les concepteurs de circuits, mais qui sont mis ici efficacement à profit (on comprend alors que la sécurité ne va pas toujours dans le même sens qu'une conception classique). Un exemple de phénomène physique que l'on trouve dans les circuits numériques est le jitter des horloges. Il s'agit d'un bruit de phase provoqué par l'accumulation de plusieurs sources de bruit dans le semi-conducteur[3]. Des montages simples, mettant en œuvre des circuits d'asservissement de fréquence par verrouillage de phase (PLL, *Phase-Locked Loop*), permettent de réaliser des générateurs efficaces de nombres aléatoires [FIS 04].

3.3.3. *Solutions matérielles de sécurité au niveau logique*

A ce niveau, les portes logiques (ET, OU, OU-EXCLUSIF, NON,…) sont considérées. En ce qui concerne la sécurité, la caractéristique essentielle est de construire des portes ne laissant échapper aucune information sur des canaux cachés. Ainsi, il devient de plus en plus complexe de mettre en œuvre des attaques matérielles par analyse.

Plusieurs techniques ont été développées dans ce sens ces dernières années. L'une des techniques majeures consiste à créer une logique duale (DPL, *Dual-rail Pre-charge Logic*) [TIR 04]. Avec cette technique, lorsqu'un transistor T_A commute de l'état bloqué à saturé[4], un transistor dual T_{A_DUAL} commute de l'état saturé à bloqué et inversement. Ainsi, la consommation dynamique de l'ensemble (T_A, T_{A_DUAL}) est toujours la même. Effectivement, à chaque commutation de T_A il y a toujours un transistor qui commute de l'état bloqué à l'état saturé et un transistor qui commute de l'état saturé à l'état bloqué. Cependant, on comprend facilement que la consommation de puissance dynamique est doublée. De plus l'augmentation du nombre de transistors du circuit conduit inévitablement à l'augmentation des fuites et donc de la consommation de puissance statique du circuit.

Cette technique a été améliorée. Par exemple, dans [POP 06] les auteurs proposent de rendre la consommation aléatoire plutôt que de la rendre constante.

---

[3] Trois bruits provoquent l'apparition du jitter. Le bruit de diffusion, provoqué par les interactions entre les électrons dans le circuit et le réseau cristallin, est dû aux mouvements aléatoires des porteurs de charge. Le bruit en excès (comme le bruit de Flicker) est un bruit en 1/f dû, entre autre, à la variation de la conductivité des matériaux. Le bruit de jonction (comme le bruit de grenaille) est dû à la traversée d'une barrière de potentiel par les porteurs de charges.

[4] Un transistor bloqué peut être grossièrement modélisé par un interrupteur ouvert, un transistor saturé par un interrupteur fermé.

Leur technique (MDPL, *Masked DPL*) consiste à mixer la technique DPL avec une technique de masquage aléatoire.

Les technologies basées sur la logique asynchrone ont des propriétés propres qui les prémunissent contre les attaques par analyse de la consommation de puissance (DPA) et les attaques par injection de fautes (DFA) [MON 06]. Cependant, il existe aujourd'hui peu de circuits facilement utilisables sur le marché pour y intégrer des systèmes complets. La conception de systèmes asynchrones est complexe et les flots de conception automatisés présentent de sévères lacunes les rendant difficilement exploitables pour une utilisation industrielle. Cependant, certaines équipes de recherche en France, cherche à faire avancer ce domaine, c'est le cas des participants du projet SAFE qui développent un circuit FPGA en logique asynchrone [SAFE].

3.3.4. *Solutions matérielles de sécurité au niveau physique*

A ce niveau, les transistors et les processus physiques de fabrication sont considérés afin de protéger physiquement le composant, et la conception. Il est nécessaire de mettre en œuvre des techniques matérielles afin d'augmenter la résistance aux attaques [AND 01]. Il est aussi essentiel d'imaginer des capteurs permettant l'analyse de l'état du composant afin de prévenir et détecter les attaques [CRA 02].

Les processus de fabrication des circuits électroniques sont complexes et de plus en plus délicats avec la réduction des technologies (aujourd'hui les technologies les plus fines sont les 65nm et 45nm). Cela entraîne de légères variations lors de la fabrication, mais celles-ci suffisent à identifier clairement un circuit d'un autre, même si ces deux circuits étaient voisins sur un disque de silicium (*wafer*) lors de la fabrication. Ces différences sont faciles à mesurer sur les lignes de connexions entre les transistors. Cette caractéristique des circuits intégrés est utilisée pour fabriquer des systèmes d'indentification et d'authentification de circuits bien utiles dans les systèmes de sécurité. Des fonctions, appelées PUF (*Physical-Random Unclonable Function*), sont alors utilisées [GAS 03].

La section suivante va présenter des architectures de systèmes sécurisées qui mettent en œuvre certaines des solutions que nous venons de présenter.

4. Architectures matérielles sécurisées pour les systèmes embarqués.

Afin d'illustrer l'utilisation des solutions matérielles détaillées dans les parties précédentes, cette section présente quelques architectures mettant en œuvre ces solutions pour trois grands types d'objectifs de protection. La classification orientée vers les objectifs n'est pas évidente car certaines solutions peuvent être utilisées

pour répondre à plusieurs problèmes de sécurité. Cependant, les mécanismes et techniques de sécurité mises en œuvre pour chacune des architectures présentées répondent principalement à un ensemble d'objectifs. C'est pourquoi nous avons choisi ce classement. Les trois types d'objectifs proposés sont :

- La protection du système d'exploitation et du logiciel embarqué contre les attaques logicielles (virus, chevaux de Troie …), et la protection des données embarquées.

- La protection de la propriété intellectuelle (DRM, protection de la conception).

- La protection des communications et applications de sécurité.

Ce chapitre n'a pas pour ambition d'être un état de l'art exhaustif de toutes les architectures sécurisées, industrielles ou académiques, disponibles aujourd'hui. Il présente des systèmes intéressants qui permettront au lecteur d'avoir une vue d'ensemble des solutions actuelles.

4.2 Architectures de protection du logiciel et des données embarqués

Les utilisateurs de systèmes programmables souhaitent bénéficier d'un système ouvert avec des propriétés de flexibilité et de généricité afin de s'adapter facilement à un large spectre d'applications. Cependant, ces mêmes utilisateurs souhaitent bénéficier dans un même temps de mécanismes de protection qui restreignent l'accès aux données sensibles et de mécanismes d'authentification qui assurent l'intégrité des données. Comme nous l'avons décrit précédemment, le système une fois embarqué peut être menacé par des attaques logicielles comme matérielles.

Un certain nombre de systèmes programmables sécurisés sont aujourd'hui proposés dans les milieux académiques et industriels. Cette sous-section propose l'étude de quelques solutions qui répondent à la problématique de sécurité des données. Ces données sont de deux sortes, les données liées à l'utilisateur et les données liées à l'application. Pour l'utilisateur les données sensibles sont, par exemple, ses données confidentielles, ses mots de passe, ses clés de chiffrement. Pour l'application il s'agit principalement de protéger l'accès au code (instructions) du programme pour limiter les attaques logicielles. Dans tous les cas, le système sécurisé doit être capable de générer, protéger et partager avec le monde extérieur les données secrètes.

Nous allons rapidement étudier les principes des attaques logicielles courantes, avant de présenter quelques solutions de protection matérielle proposées dans la littérature. Pour cela nous considérons que le système d'exploitation n'est pas sûr et

qu'il peut être affecté par un code malicieux. Une fois que l'attaquant a le contrôle du système d'exploitation, il peut avoir tous les privilèges d'accès aux mémoires (y compris à la pile), ce qui lui permet de les observer et de les modifier. De plus, il peut avoir le contrôle des interruptions ce qui lui permet d'avoir accès aux valeurs des registres. L'attaquant peut alors positionner une valeur aléatoire sur le bus pour causer un disfonctionnement et observer le comportement du système, c'est une attaque par injection de faute logicielle (en anglais *Spoofing Attack*). Il peut modifier, par permutation des adresses, le contenu de la mémoire d'instructions. De cette façon il est possible de modifier, par exemple, l'adresse de retour d'un sous-programme (ou d'un programme d'interruption). C'est une technique d'attaque par permutation spatiale d'instructions (en anglais *Splicing Attack*). Enfin, l'attaquant peut modifier la mémoire de données pour donner une ancienne valeur à une donnée. Ainsi, la donnée sera utilisée plusieurs fois avec la même valeur qui est alors erronée. On parle alors d'attaque par rejeu (en anglais *Replay Attack*).

Il est donc nécessaire que l'attaquant ne puisse pas changer les instructions et les données en mémoire sans que le processeur s'en aperçoive. De plus, il est nécessaire que ces mêmes instructions et données ne soit pas compréhensibles en dehors du processeur. Les solutions proposées aujourd'hui s'appuient sur une zone de sécurité matérielle inviolable (*Trust Zone*, *Trust Area* ou *Secure Area* en anglais) qui contient le processeur, la ou les mémoires caches et le contrôleur d'accès à la mémoire. Cette zone met en œuvre les systèmes de protection matérielle au niveau physique, logique et architectural afin de se prémunir des attaques matérielles connues. Les données qui sortent et qui rentrent dans cette zone de sécurité sont chiffrées afin d'assurer la protection au niveau système. La zone de sécurité contient les primitives matérielles permettant le chiffrement et l'authentification des données (instructions, adresses, données) qui sont échangées entre le processeur et les différentes mémoires.

L'architecture **CryptoPage-2** proposée par l'ENST Bretagne [LAU 03] et l'architecture du processeur sécurisé **AEGIS** développée par les chercheurs du MIT aux USA [SUH 03] sont très proches de la description que nous venons de faire d'une architecture de protection du logiciel. Le schéma de l'architecture **AEGIS** est donné à la figure 3, sur celui-ci les zones sécurisées et non-sécurisées sont clairement séparées. De plus, une deuxième version de ce processeur propose d'utiliser des fonctions physiques aléatoires, PUF, qui tirent partie des faibles différences de longueurs de fils microélectroniques dues aux variations lors de la fabrication, afin de fournir un identifiant unique par circuit. Ces identifiants sont utilisés pour générer des clés de chiffrement uniques [SUH 05].

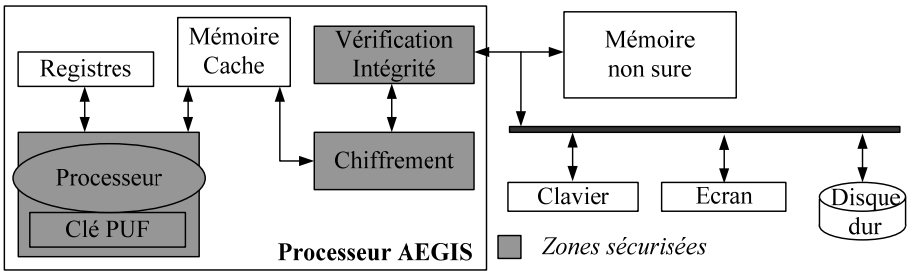

Figure 3. *Architecture du processeur sécurisé AEGIS.*

Ces architectures sont proches de l'architecture **XOM (*Execute-Only Memory*)** proposée par l'Université de Stanford [LIE 00]. La différence principale tient dans le fait que chaque bloc d'instructions, correspondant à une tâche à exécuter sur ce processeur, est chiffré à l'aide d'un algorithme de chiffrement à clé symétrique, chaque clé symétrique est chiffrée avec un algorithme de chiffrement asymétrique. Pour lire un bloc d'instructions, le processeur **XOM** doit tout d'abord déchiffrer la clé symétrique correspondante, puis déchiffrer les instructions. Cette méthode, bien que plus sécurisée, est plus lente et consomme plus de ressources silicium que l'utilisation d'un seul chiffrement symétrique avec sauvegarde de la clé dans la partie sécurisée. Effectivement, il faut que la zone sécurisée embarque deux unités de chiffrement/déchiffrement différentes et qu'elle ait des zones mémoires suffisantes pour stocker les résultats des déchiffrements successifs.

La gestion des clés dans la partie sécurisée est un sous système souvent complexe qui demande à être développé avec soin car toute la sécurité de l'algorithme de chiffrement tient dans la protection des clés utilisées. Cette partie est particulièrement soignée dans l'architecture de processeur sécurisé ***TSM (Trusted Software Module)*** proposée par l'Université de Princeton aux USA [LEE 05], schématisée à la figure 4. La gestion des différentes clés (dont le nombre peut être très élevé) dans l'architecture sécurisée est faite de façon hiérarchique. Chaque clé utilisée est calculée à partir d'une clé parent. La clé de plus haut niveau n'a pas de parent, elle est la clé maître (*User Master Key*). Cette clé doit être particulièrement sûre puisque toutes les autres y sont liées hiérarchiquement. La clé maître est associée à l'utilisateur et non au circuit comme dans le cas d'**AEGIS**. Le couplage clé-utilisateur peut se faire via des informations biométriques. Les clés peuvent êtres utilisées par les périphériques du processeur **TSM** sauf la clé maître qu'il est le seul à pouvoir lire et utiliser.

Le projet **PE-ICE** développé à l'Université de Montpellier [ELB 06], propose de renforcer la sécurité de système tel que **AEGIS** en ajoutant un bloc d'authentification. Celle-ci ce fait en ajoutant à la donnée à chiffrer quelques bits supplémentaires avant le chiffrement. Ces bits sont calculés à partir d'une valeur

aléatoire et de l'adresse de stockage de la donnée en mémoire (afin d'éviter les réallocations spatiales de données en mémoire). Le principal inconvénient de cette méthode est l'augmentation de la taille des données à stocker en mémoire.

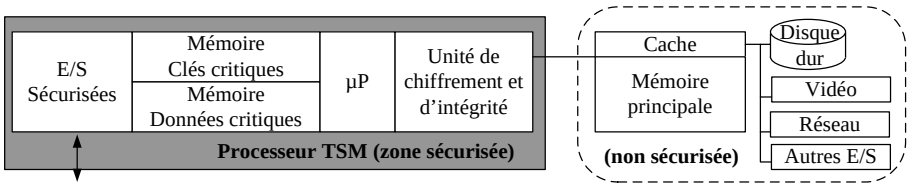

Figure 4. *Structure de l'architecture TSM (Trusted Software Module).*

Les solutions présentées ci-dessus sont efficaces pour remplir les objectifs de sécurités mais elles entrainent une diminution des performances. Les temps liés au chiffrement et déchiffrement des instructions et des données ne sont pas négligeables par rapport au temps normalement nécessaire à la lecture ou l'écriture de données en mémoire. Pour répondre à ce problème, le projet **OTP-CRC** (*One-Time Pad and Cyclic Redundancy Code*) de l'Université de Bretagne Sud [VAS 07], propose de réaliser le chiffrement des données par le simple ajout d'une clé unique générée à partir d'un algorithme de chiffrement symétrique (AES). Un système de détection d'erreur classiquement utilisé en communication par bus (CRC) est utilisé pour vérifier l'intégrité des données déchiffrées. Concernant la réduction de la taille de la mémoire (données ou instructions) la même équipe de recherche propose de mixer un algorithme de compression par dictionnaire avec la phase de chiffrement [WAN 07].

Les architectures de processeurs proposées sont sécurisées mais relativement complexes à mettre en œuvre. Pour des systèmes embarqués industriels avec de fortes contraintes d'intégration, il est nécessaire de disposer d'un processeur sécurisé plus simple et plus performant. C'est ce que propose la société ARM, dont les processeurs sont largement utilisés dans le monde de l'embarqué, avec son extension **TrustZone** [ARM 07]. Ce système de contrôle du processeur entraine une augmentation de 15.10^3 à 20.10^3 du nombre de portes logiques nécessaires. Cependant, cela représente seulement 5% de silicium supplémentaire pour un cœur de processeur ARM11. Lorsqu'une attaque logicielle est repérée, à l'aide des contrôles des privilèges (contrôle des accès en lecture et écriture aux mémoires), le système **TrustZone** fait passer le processeur d'une configuration normale à une configuration sécurisée.

Ces différents processeurs sécurisés peuvent être utilisés dans de nombreuses applications nécessitant différents niveaux de sécurité, cependant les applications de protection de la propriété intellectuelle peuvent nécessiter des systèmes de sécurité supplémentaires comme nous allons le voir dans la suite de ce chapitre.

4.2. Architectures de protection de la propriété intellectuelle

La protection de la propriété intellectuelle est un large domaine dans celui de la sécurité. L'expression *protection de la propriété intellectuelle*, regroupe des applications très différentes. Par exemple la gestion des droits numériques (DRM ou *Digital Rights Management*) pour les supports multimédia ou la protection de la conception dans le cadre de l'espionnage industriel. Dans ce dernier cas il convient de protéger la conception du système contre la copie et l'ingénierie inverse.

Dans le cas de la conception de systèmes industriels, il existe un autre problème de sécurité et de confidentialité de la conception. Les systèmes électroniques actuels, du fait de l'augmentation constante de la complexité, nécessitent un développement conjoint entre plusieurs acteurs (sous-traitants, fournisseur, client). Chacune des entités peut souhaiter que la partie du système qu'elle développe soit vue comme une boite noire par les autres entités. Bien qu'il convienne tout d'abord de mettre en place un dispositif juridique pour prévenir les problèmes, comme des accords de confidentialité (NDA ou *Non Disclosure Agreement*), certaines solutions techniques aident à consolider ce dispositif. C'est le cas de la solution **CodeGuard** proposée par la société Microchip dans certains de ces microcontrôleurs [MIC 07]. Cette solution matérielle permet, par un contrôle des registres internes du microcontrôleur, de vérifier les privilèges en lecture et écriture sur les mémoires d'instructions et de données. Chaque entité (dont le nombre est ici limité à trois) participant à la conception peut se voir attribuer un espace mémoire et un niveau de sécurité. Il existe trois niveaux de sécurité, du niveau le plus sûr très restreint en lecture et écriture au niveau le moins sûr sans restriction. Cette solution, quoique assez limitée, permet le développement conjoint d'une application sur un système programmable en respectant les informations de conception des différentes entités.

L'architecture **SECA (*Security-Enhanced Communication Architecture*)** est une architecture de système sur puce (SOC ou *System On Chip*), pour des applications en téléphonie mobile, centrée autour d'un contrôleur d'adresses, de données et de transactions sur un bus de type AMBA[5]. Cette architecture est proposée par le laboratoire américain de NEC [COB 05] pour des applications du type DRM. Elle dispose de trois types de protection via une unité de contrôle spécifique reliée à un contrôleur global le SEM (*Security Enforcement Module*). Ce module directement connecté au bus AMBA contrôle les échanges de données entre le ou les processeurs, les différentes mémoires (instructions, données), les périphériques. Ce module assure le contrôle des privilèges d'accès pour un composant vers un espace adressable de la mémoire ou vers un périphérique

[5] AMBA est un standard libre de communication sur bus pour les systèmes sur puce développé par la société ARM.
http://www.arm.com/products/solutions/AMBAHomePage.html

(*address-based protection*). Il contrôle, de plus, les valeurs des données qui sont accessibles par certaines parties de la mémoire ou par certains registres des périphériques (*data-based protection*). Enfin, les séquences de transactions entres les différents composants de l'architecture sont contrôlées afin de vérifier le comportement normal du système (*sequence-based protection*).

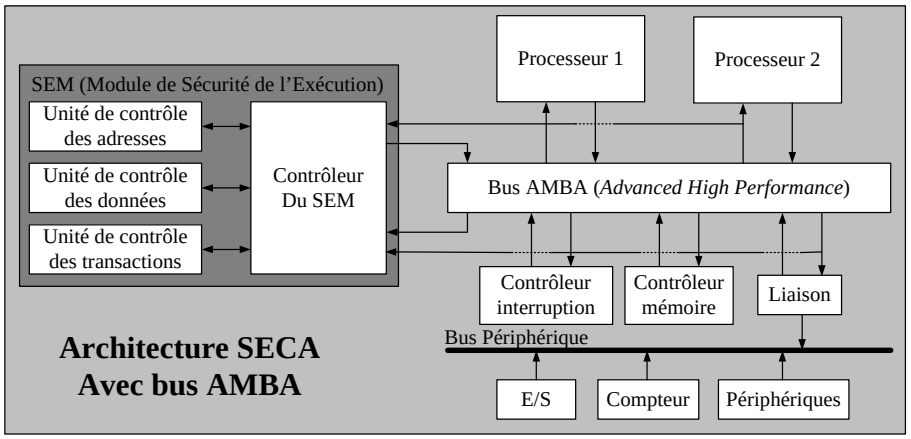

Figure 5. *Un exemple d'architecture SECA avec deux processeurs.*

Le contrôleur de sécurité, SEM, comme on le voit sur le schéma général de l'architecture **SECA** proposé à la figure 5, est le composant central du contrôle des communications. Afin de contrôler les adresses, les données et les séquences d'accès tout en gardant une complexité restreinte, le contrôleur utilise trois unités séparées. Une unité de contrôle des adresses (*Addresse-based Protection Unit*) basée sur une table de privilèges d'accès en lecture et écriture, une unité de contrôle des données manipulées (*Data-based Protection Unit*) basée sur une LUT (*Look Up Table)* qui vérifie le niveau d'accès aux données pour chaque composant. Enfin, l'unité de contrôle des transactions (*Sequence-based Protection Unit*) est basée sur une machine à états finis, construite à partir d'une étude du comportement normal de fonctionnement.

Cette architecture est flexible, le contrôleur de sécurité peut être limité par exemple au contrôle des données manipulées comme dans l'application dans le SOC NEC MP211 pour la téléphonie mobile [COB 05]. Cette architecture peut être alors efficacement mise en œuvre pour la protection des droits numériques des applications multimédia.

Les solutions apportées par les systèmes **CodeGuard** et **SECA** répondent aux problématiques des systèmes programmables centrés autour d'un ou plusieurs microprocesseurs. Cependant, dans un souci de performance, les systèmes matériels

sont de plus en plus utilisés. Ils sont centrés autour de circuits spécialisés comme les ASIC ou autour de circuits reconfigurables comme les FPGA. Dans ce dernier cas la conception du système tient dans un fichier de configuration appelé *bitsteam*. Si un concurrent peut facilement extraire ce fichier du système, il pourra le copier voir le comprendre à l'aide des procédés d'ingénierie inverse. Or, les FPGA de technologie SRAM ont un problème critique de sécurité. La technologie de sauvegarde de la configuration dans le circuit est la technologie de mémoire volatile SRAM. Afin de ne pas perdre la configuration à chaque coupure de l'alimentation en énergie, le *bitstream* doit être stocké dans une mémoire du type FLASH ou ROM externe. Ainsi, à chaque mise sous tension le système charge le *bitstream* depuis la mémoire externe non volatile vers la mémoire de configuration interne du FPGA. Il est alors aisé pour un attaquant de lire le *bitstream* lors de ce transfert. Pour pallier à cette faille de sécurité, les fabricants de FPGA SRAM, comme Xilinx ou Altera, proposent de stocker le bitstream chiffré dans la mémoire externe et de le déchiffrer à l'intérieur du FPGA (grâce à un circuit de déchiffrement embarqué). Cette solution, bien que simple à mettre en œuvre, est très figée et ne laisse que peu de choix au développeur. Cependant des études ont montré qu'il est possible de fournir des services de protection du bitstream plus complets et plus flexibles [BOS 04b].

La conception de systèmes matériels s'appuie de plus en plus sur l'utilisation de composants virtuels (IP ou *Intellectual Properties*) dans un souci d'efficacité (réduction du *time to market*). La vente de ces IP représente aujourd'hui un important marché qui est lui aussi soumis à des problématiques de sécurités. La protection des IP est un enjeu important pour le développement de ce marché. Son rôle est de permettre aux vendeurs de protéger leurs IP contre toutes utilisations non autorisées, ou la revente frauduleuse en assurant une traçabilité de l'IP pour des besoins juridiques. Des techniques de marquages (*watermaking*, *fingerprinting*) sont mises en œuvre pour répondre à ces besoins. Parmi les nombreux exemples de techniques de marquages des IP matériels, on peut citer le changement de taille des lignes de routage pour ajouter une information physique indétectable, l'utilisation de ressources de silicium libres (comme les LUT libres dans un FPGA configuré) ou la modification des paramètres de l'algorithme. C'est un vaste sujet qui mériterait à lui seul un chapitre. C'est pourquoi les auteurs renvoient les lecteurs intéressés vers les références suivantes [VSI 06], [ABD 04] et [YUA 06].

4.3 Crypto-architecture pour la protection des communications et les applications de sécurité

Les architectures présentées précédemment intègrent des systèmes (ou primitives) de sécurité (algorithmes de chiffrement par exemple) qui sont utilisés en interne pour la protection des informations sensibles (clés, instructions, données). Dans le cas des crypto-processeurs (ou crypto-coprocesseurs) les primitives de sécurités sont utilisées dans le cadre d'applications de sécurité comme les cartes à

puces, des télécommunications et protocoles sécurisés (IPsec par exemple), les réseaux virtuels privés (VPN ou *Virtual Private Network*). Le plus souvent sous la forme de coprocesseurs, les crypto-processeurs embarquent généralement, suivant l'application ciblée, des primitives de chiffrement (symétriques ou non), des fonctions de hachage pour l'authentification, des générateurs de clés basés sur des générateurs de nombres aléatoires et un système de sauvegarde et de gestion des clés. L'enceinte physique du crypto-processeur est sécurisée contre les attaques matérielles. Dans certains systèmes, des capteurs sont embarqués pour surveiller l'environnement interne et externe du circuit et détecter des attaques.

Les premiers processeurs de sécurité, développés suivant ce principe, visaient principalement les applications de sécurité des réseaux. IBM est un des principaux concepteurs de crypto-processeurs pour ces applications (comme le processeur 4764 PCI-X[6]). Ces processeurs sont principalement installés dans des serveurs pour lesquels les contraintes de développement et de performance sont loin de celles des ordinateurs de bureaux ou mobiles. C'est pourquoi un consortium s'est monté pour développer un standard de crypto-processeurs pour ces cibles. Le **TCG** (*Trusted Computing Group*) regroupe des industriels du secteur de l'informatique dans le but de définir des standards ouverts qui répondent aux besoins des applications de sécurité [TCG 07].

Parmi ces différents travaux, le **TCG** propose une architecture de crypto-coprocesseur appelée **TPM** (*Trusted Platform Module*) [TCG 07]. Un schéma bloc de cette architecture est donné à la figure 6. Sur celle-ci, on peut voir, à droite, les primitives de sécurité qui sont utilisées en fonction des instructions à exécuter. On peut modéliser ces primitives par une super-ALU spécialisée, utilisée pour réaliser des opérations nécessaires aux applications de sécurité. Bien sûr, l'utilisation de telles architectures nécessite le développement d'un système d'exploitation et de logiciels capables de les utiliser efficacement.

Le **TCG** a depuis peu lancé une réflexion pour développer un standard du type **TPM** pour les systèmes embarqués mobiles et sans fils (principalement les téléphones portables, les assistants électroniques et les ordinateurs ultra-portables). Cependant des crypto-processeurs pour systèmes embarqués sont déjà développés et commercialisés. Texas Instrument propose, pour la troisième génération de son processeur **OMAP** (TI OMAP 3430[7]) pour les applications mobiles embarquées, un coprocesseur sécurisé. Celui-ci est basé sur la technologie ARM TrustZone [ARM 07]. Ce processeur embarque des fonctions de chiffrements symétriques (AES, DES et triple-DES), des fonctions de hachage (SHA-1, MD5), un générateur de nombres aléatoires et un système de gestion des clés. Dans le même ordre d'idée, la société

[6] http://www-03.ibm.com/security/cryptocards/pcixcc/overview.shtml
[7] hhtp://www.ti.com/omap

Discetix a développée le processeur **CryptoCell**[8] pour les applications mobiles. Ce coprocesseur embarque des fonctions de chiffrements asymétriques (RSA, ECC et DH), des fonctions de chiffrements symétriques (AES, DES, triple-DES, RC4), des fonctions de hachage (SHA-1, SHA256/384/512/ MD5, HMAC), un générateur de nombres aléatoires et un système de gestion des clés.

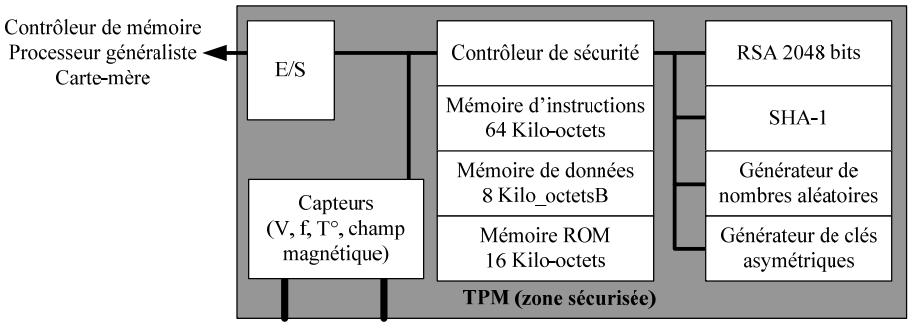

Figure 6. *Architecture des TPM (Trusted Platform Module).*

L'offre existe donc pour le développement de systèmes embarqués capables de supporter des applications de sécurité. Cependant, les solutions proposées aujourd'hui répondent uniquement aux besoins actuels. Dans une perspective à plus long terme, il est indispensable de proposer de nouvelles architectures capables d'évoluer dans le temps (mise à jour des algorithmes) tout en assurant des performances élevées. L'architecture **SANES** présentée dans la section suivante est une de ces nouvelles architectures évolutives.

4.4. Un cas d'étude : SANES une architecture matérielle sécurisée reconfigurable

L'architecture reconfigurable **SANES** (*Security Architecture for Embedded Systems*) a été développée conjointement par l'Université de Bretagne Sud et par l'Université du Massachusetts aux USA [GOG 06]. Cette architecture correspond à la mise en œuvre des principes définis au sein de la vision conceptuelle (vue au chapitre 3.3) en une vision architecturale. Cette architecture utilise des moniteurs qui permettent de détecter les comportements anormaux du système. Des mécanismes matériels de défense peuvent alors être mis en œuvre afin de contrer les attaques. Par ailleurs, les mécanismes de sécurité peuvent être mis à jour si besoin (de façon dynamique) ce qui assure la pérennité du système de protection.

[8] http://discretix.com/CryptoCell/

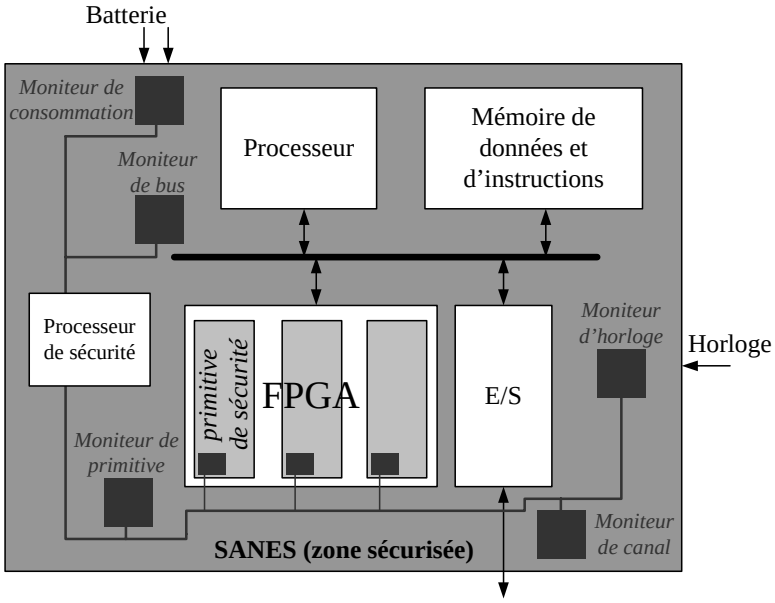

Figure 7. *L'architecture reconfigurable sécurisée SANES.*

La figure 7 présente une vue générale de l'architecture. Comme on peut le voir sur cette figure, plusieurs moniteurs sont considérés afin de surveiller différentes sources d'information du système. Le nombre et la complexité des moniteurs sont bien évidemment des paramètres importants car ils conditionnent directement les surcoûts liés à l'architecture de sécurité ainsi que le niveau de sécurité. Le rôle de ces moniteurs est de détecter les attaques contre le système. Pour cela, l'activité normale des modules sous surveillance est caractérisée et comparée en permanence avec l'activité réelle du système. Les notions d'autonomie et d'adaptabilité des moniteurs sont importantes afin de construire un réseau de surveillance efficace. En effet, les moniteurs sont autonomes afin de correspondre à un système tolérant aux fautes; si un moniteur est attaqué, les autres doivent être en mesure de continuer à assurer la sécurité du système. Les moniteurs sont distribués à différents endroits stratégiques du système afin d'analyser les points faibles de l'architecture (e.g. batteries, bus, primitives de sécurité, canaux de communication).

Différents niveaux de réaction peuvent être considérés en fonction du type d'attaques auxquelles le système doit faire face. Les réactions de type réflexes sont réalisées directement par un moniteur sans concertation avec les autres unités de sécurité. Dans ce cas le temps de réaction est très rapide. Les réactions de type globales sont mises en œuvre lorsqu'une attaque implique une modification importante du système. Dans ce cas, les moniteurs échangent des informations afin

de définir une nouvelle configuration. Un tel scénario permet de détecter des attaques plus complexes mais implique également un temps de réaction plus long. Les moniteurs sont connectés par un réseau sur silicium sécurisé. Ce réseau est également connecté à une unité de contrôle globale appelée SEP (*Security Executive Processor*) dont le rôle est d'assurer le lien sécurisé entre l'environnement extérieur et le système. Le contrôleur SEP correspond à une couche logicielle permettant d'instancier à distance de nouveaux moniteurs et de mettre à jour les politiques de sécurité des moniteurs existants. En cas de comportement anormal, le contrôleur SEP peut prendre le contrôle du système du point de vue matériel. Il peut par exemple annuler la gestion du niveau de batterie ou déconnecter des entrées/sorties afin de faire échouer une attaque.

La partie reconfigurable (FPGA) au sein du système permet l'implantation matérielle des primitives de sécurité. Cela conduit à disposer d'un accélérateur matériel adaptatif réalisant un algorithme lié à la sécurité (chiffrement, hachage, gestion des clés). Contrairement aux crypto-processeurs présentés précédemment, la liste des algorithmes supportés n'est pas figée. L'utilisateur configure le système avec les primitives de sécurité qu'il souhaite embarquer (et dont il dispose). Celles-ci pouvant être mises à jour par reconfiguration durant la vie du système. Ainsi, l'architecture **SANES** apporte la performance (implantation matérielle) et la flexibilité (système reconfigurable) nécessaire aux futurs systèmes embarqués sécurisés.

6. Conclusion

Les systèmes embarqués sont au cœur d'un marché économique important qui est un des moteurs de l'économie des nouvelles technologiques. Cependant, ces systèmes deviennent de plus en plus complexes, mobiles et communicants, ce qui les rend de plus en plus vulnérables aux problèmes de sécurité, que ce soit la sécurité des données, du système ou de la conception.

Fortement contraint, le développement des systèmes embarqués empêche l'utilisation directe des solutions de sécurité (logicielles et matérielles) disponibles aujourd'hui et développées pour d'autres cibles (cartes à puce, ordinateurs de bureaux ou portables, serveurs). Il est donc indispensable de développer des solutions adaptées aux systèmes embarqués en adéquation avec leurs spécificités et qui respectent les contraintes de développement.

Aujourd'hui de nombreuses solutions, académiques et industrielles, sont proposées pour répondre à ce challenge. Cependant un certain nombre d'efforts de recherche est encore nécessaire aujourd'hui. Principalement, pour les plateformes il s'agit de les rendre plus flexibles sans réduire pour autant leurs performances,

d'augmenter la protection globale du système en restant dans un coût (financier ou technologique) raisonnable. Pour les outils, il est nécessaire de développer des flots de conception automatique intégrant dès les premières phases de spécification la contrainte de sécurité. Ces flots doivent s'appuyer sur de nouvelles méthodes de conception sécurisées qui restent à développer.

Enfin, un dernier point reste à considérer : la formation des ingénieurs de développement et de recherche pour le domaine des systèmes embarqués doit prendre un compte les aspects sécurités. Si ceux-ci font depuis longtemps parti du cursus classique en informatique ou en réseau, ce n'est pas encore tout à fait le cas dans les cursus en électronique embarquée. Cependant, un certain nombre d'initiatives en France et à l'étranger prouve que cela est en train de changer.

7. Bibliographie

[ABD 04] A.T. Abdel-Hamid, S. Tahar, M. Aboulhamid, « A Survey on IP Watermarking Techniques », *in Design Automation for Embedded Systems*, pp. 211-227, 2004.

[AND 01] R. Anderson, *Security Engineering, A Guide to Building Dependable Distributed Systems*, Wiley Computer Publishing, ISBN 0-471-3892-6, 2001.

[ARM 07] http://www.arm.com/products/esd/trustzone_home.html

[ARO 05] D. Arora, S. Ravi, A. Raghunathan, and N. K. Jha, « Secure Embedded Processing through Hardware-assisted Run-time Monitoring », *in Proceedings of Design, Automation & Test in Europe Conference (DATE 2005)*, mars 2005.

[BER 03] G. Bertoni, L. Breveglieri, I. Koren, P. Maistri, and V. Piuru, « Error Analysis and Detection Procedure for a Hardware Implementation of the Advanced Encryption Standard », *in IEEE Transactions on Computers*, Vol. 52, N° 4, pp. 492-505, avril 2003.

[BOS 04a] L. Bossuet, *Exploration de l'espace de conception des architectures reconfigurables*, Thèse de doctorat, Université de Bretagne Sud, Lorient, septembre 2004, disponible librement à l'adresse http://www.lilianbossuet.com/fr/Doc/publications/These_Lilian_Bossuet.pdf

[BOS 04b] L. Bossuet, G. Gogniat, and W. Burleson, « Dynamically Configurable Security for SRAM FPGA Bitstreams », *in Proceedings of the 11th Reconfigurable Architectures Workshop (RAW 2004)*, Santa Fé, New Mexico, USA, avril 2004.

[BOS 06] L. Bossuet, *Architecture Conception et Utilisation des FPGA*, Cours de l'ENSEIRB 2006, disponible librement à l'adresse http://www.lilianbossuet.com/fr/Doc/documents_pedagogiques/Bossuet_cours_FPGA_ENSEIRB.pdf

[BUR 05] W. Burleson, T. Wolf, R. Tessier, W. Gong, G. Gogniat, « Embedded System Security: A Configurable Approach », *Department of Homeland Security Conference*, Boston, Massachusetts, USA, avril 2005.

[CHO 05] H. Choukri and M. Tunstall, « Round Reduction Usign Faults », in L. Breveglieri and I. Koren, Eds., *Workshop on fault Diagnosis and Tolerance in Cryptography (FDTC 2005)*, pp. 13-24, 2005.

[COB 05] J. Coburn, S. Ravi, A. Raghunathan, S. Chakradhar, « SECA: Security-Enhanced Communication Architecture », *In Proceeding of International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES'05)*, San Fransisco, USA, septembre 2005.

[CRA 02] N. Cravotta « Prying eyes », *EDN*, sptembre 2002, http://www.edn.com/toc-archive/2002/20020926.html

[DAE 02] J. Daemen and V. Rijmen, *The Design of Rijndael AES-The Advanced Encryption Standard*, Springer-Verlag, 2002.

[DAG 04] D. Dagon, T. Martin, and T. Staner, « Mobile Phones as Computing Devices: The Viruses are Coming! », *IEEE Pervasive Computing*, octobre-décembre 2004.

[EVA 05] S. Evain and J.P. Diguet, « From NoC Security Analysis To Design Solutions », *IEEE 2005 Workshop on Signal Processing Systems (SIPS 2005)*, Athens, Greece, novembre 2005.

[ELB 06] R. Elbaz, *Mécanismes Matériels pour des transferts Processeur Mémoire Sécurisés dans les Systèmes Embarqués*, Thèse de doctorat, Université de Montpellier, décembre 2006.

[FIS 04] V. Fischer, M. Drutarovský, M. Šimka, and N. Bochard, « High Performance True Random Number Generator in Altera Stratix FPLDs ». In J. Becker, M. Platzner, and S. Vernalde, editors, *Field-Programmable Logic and Applications (FPL 2004*), Vol. 3203 of Lecture Notes in Computer Science, Springer-Verlag, pp. 555–564, Antverp, Belgium, août 2004.

[FIL 06] E. Filiol, *Virus et Ver informatiques*, Chapitre 6 du Traité I2C, Sécurité des systèmes d'informations, sous la direction de L. Mé et Y. Deswarte, pp. 187-219, éditions Hermes - Lavoisier, mai 2006. ISBN 2-7462-1259-5

[GAS 03] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, « Delay-Based Circuit Authentication and Applications ». In *Proc. of the 18th Annual ACM Symposium on Applied Computing*, mars 2003.

[GIR 03] C. Giraud, *DFA on AES*, Technical Report 2003/2008, IACR eprint archive, 2003. http://eprint.iacr.org/2003/008.ps.

[GOG 06] G. Gogniat, T. Wolf, W. Burleson, « Reconfigurable security support for embedded systems », In *Proc of the 39th IEEE Hawaii International Conference on System Science (HICSS-39)*, Poipu, HI, USA, janvier 2006.

[GUI 04] S. Guilley and R. Pacalet, « SoC Securiy: a War Against Side-Channels », *Annals of the Telecommunications*, Système sur puce électronique pour les télécommunications Vol. 59, N° 7-8, juillet-août 2004.

[HAV 00] P. J.M. Havinga, G. J.M. Smit, « Design techniques for low power systems », *Journal of Systems Architecture*, Vol. 46. Iss 1, 2000.

[ICTER 06] http://www.lirmm.fr/~w3mic/ANR/index.htm

[KAR O2] Karri, K. Wu, P. Mishra, and Y. Kim, « Concurrent Error Detection Schemes for Fault-Based Side-Channel Cryptanalysis of Symmetric Block Ciphers », *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 21, N° 12, décembre 2002.

[KOC 96] P. Kocher, « Timing Attacks on Implementations of Diffe-Hellman, RSA, DSS, and Other Systems, Advances in Cryptology », *in Proceedings of Annual International Cryp-tology Conference (CRYPTO '96)*, Springer-Verlag, pp. 104-113, août 1996.

[KOC 99] P. C. Kocher, J. Jaffe, and B. Jun, « Differential Power Analysis », In Michael Wiener, editor, *in Proceedings of the 19th Annual International Cryptology Conference (CRYPTO'99)*, Vol. 1666 of Lecture Notes in Computer Science, pp. 388-397, Springer, Santa Barbara, California, USA, août 1999.

[KOC 04] P. Kocher, R. Lee, G. McGraw, A. Raghunathan, and S. Ravi, « Security as a New Dimension in Embedded System Design », *ACM/IEEE Design Automation Conference*, juin 2004.

[LAU 03] C. Lauradoux, R. keryell, « CryptoPage-2 : un processeur sécurisé contre le rejeu », *REMPAR'15/CFSE'3/SympAAA'2003*, octobre 2004.

[LEE 05] R.B. Lee, P.C.S. Kwan, J.P. Mcgregor, J. Dwoskin, Z. Wang, « Architecture for Protecting Critical Secrets in Microprocessors », *In Proceedings of the 32nd International Symposium on computer Architecture (ISCA 2005)*, pp 2-13, juin 2005.

[LIE 00] D. Lie, C. Thekkath, M. Mitchell, P. Lincoln, D. Boneh, J. Mitchell and M. Horowitz, « Architectural Support for Copy and Tamper Resistant Software », *In Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS IX)*, pp. 168-177, novembre 2000.

[MEN 96] P. Menezes, V. Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press ISBN 0-8493-8523-7, octobre 1996.

[MIC 07] http://www.microchip.com/codeguard/

[MON 06] Y. Monnet, M. Renaudin, R. Leveugle, « Designing Resistant Circuits against Malicious Faults Injection Using Asynchronous Logic », *IEEE Transaction of Computer*, N°55, Vol 9, pp. 1104-1115, 2006.

[MES 01] T. Messerges, « Securing the AES finalists Against Power Analysis Attacks », *Fast Encryption Workshop (FSE 2000)*, LNCS 1978, pp. 150-164, Springer-Verlag, 2001.

[MES 05] D. Mestiqua, J.D. Techer, L. Torres, G. Cambon, G. Sassatelli, F.G. Moraes, « Current Mask Generation : An Analogical Circuit to Thwart DPA Attacks », *in International Conference on Very Large Scale Integration (VLSI-SOC'05)*, 2005.

[NBS 01] National Bureau of Standards. FIPS 197, *Advanced Encryption Standard. Federal Information Processing Standard*, NIST, U.S. Dept. of Commerce., 2001.

[NEC 01] J. Nechvatal M. Smid D.L. Banks A. Rukhin, J. Soto, « A Statistical Test Suite for Random and Pseudorandom Number Generators for Statistical Applications ». *NIST Special Publication in Computer Security*, pp. 800–22, 2001.

[POP 06] T. Popp, S. Mangard, « Implementation Aspects of the DPA-Resistant Logic Style MDPL », *in Proceedings of the 2006 IEEE International Symposium on Circuits and Systems (ISCAS 2006)*, Island of Kos, Greece, mai 2006.

[QUI 01] Jean-Jacques Quisquater and David Samyde, « ElectroMagnetic Analysis (EMA): Measures and counter-measures for smart cards », in Isabelle Attali and Thomas P. Jensen, editors, *Proceedings of E-smart*, Vol. 2140 of Lecture Notes in Computer Science, pages 200-210. Springer-Verlag, 2001.

[SAFE] *Secured Asynchronous FPGA for Embedded Systems*. http://www.comelec.enst.fr/recherche/safe/

[SCH 03] P. Schaumont and I. Verbauwhede, « Domain-Specific Codesign for Embedded Security », *IEEE Computer*, avril 2003

[SKO 02] S. Skorobogatov and R. Anderson. « Optical Fault Induction Attacks », *in proceedings of Cryptographic Hardware and Embedded Systems Workshop (CHES 2002)*, Lecture Notes in Computer Science No. 2532, pp. 2-12, 2002.

[SUH 03] G. E. Suh, D. Clarke, B. Gassend, M. van Dijk, S. Devadas, *AEGIS: Architecture for Tamper-Evident and Tamper-Resistant Processing*, MIT, Memo-461, février 2003.

[SUH 05] G. E. Suh, C.W. O'Donnell, I. Sachdev, S. Devadas, « Design and Implementation of the AEGIS Single-Chip Secure Processor Using Physical Random Functions", *in Proceddings of the 32nd Annual International Symposium on Computer Architecture (ISCA 2005)*, pp. 25-36, 2005.

[TCG 07] *Trusted Computing Groupe Web Site* www.trustedcomputinggroup.org

[TIR 04] K. Tiri and I. Verbauwhede, « A Logic Level Design Methodology for a Secure DPA Resistant ASIC or FPGA Implementation », *in Proc. of Design Automation and Test in Europe Conference (DATE 2004)*, pp. 246-251, février 2004.

[TOS 07] http://www.toshiba-europe.com/mobile/

[VAS 07] R. Vaslin, G. Gogniat, J.P. Diguet, E. Wanderley, R. Tessier, W. Burleson, « Low Latency Solution for Confidentiality and Integrity Checking in Embedded Systems with Off-Chip Memory », *in Reconfigurable Communication-centric SoCs (ReCoSoc'07)*, Monpellier, France, juin 2007.

[VSI 01] Virtual Socket Interface Alliance, Intellectual Property Protection Development Working Group, *Intellectual Property Protection : Schemes, alternatives and discussion*, White Paper, janvier 2001.

[WAN 07] E. Wanderley , R. Elbaz, L. Torres, G. Sassastelli, R. Vaslin, G. Gogniat, J.P. Diguet, « IBC-EI: An Instruction Based Compression Method with Encryption and Integrity Checking », *in Reconfigurable Communication-centric SoCs (ReCoSoc'07)*, Monpellier, France, juin 2007.

[WOL 04] T. Wollinger, J. Guajardo, and C. Paar, « Security on FPGAs: State of the Art Implementation and Attacks », *in ACM Transactions on Embedded Computing Systems*, Vol. 3, N° 3, pp. 534–574, août 2004.

[WOL 06] T. Wolf, S. Mao, D. Kumar, B. Datta, W. Burleson, and G. Gogniat, « Collaborative Monitors for Embedded System Security », *in Proceedings of the First International Workshop on Embedded Systems Security*, Seoul, Korea, octobre 2006.

[YUA 06] L. Yuan, G. Qu, L. Ghouti, A. Bouridane, « VLSI Design Ip Protections: Solutions, New Challenges, and Opportunities », *in Proceedings of the first NASA/ESA Conference on Adaptative Hardware and Systems (AHS'06)*, Istanbul, Turkey, juin 2006.