Fiches pédagogiques sur la cybersécurité au sein des composants électroniques

CyberEdu

CyberEdu

La sécurité par l'enseignement supérieur des NTIC

Ce document pédagogique a été rédigé par un consortium regroupant des enseignants-chercheurs et des professionnels du secteur de la cybersécurité.

Version 1.7 — Février 2017

Table des matières

Introduction

Les fiches pédagogiques présentées dans ce guide ont pour objectif de mettre en avant les éléments fondamentaux de la sécurité des composants électroniques qui peuvent être présentés à des étudiants de l'enseignement supérieur non spécialistes de la sécurité. Les fiches apportent à l'enseignant des repères pédagogiques mais ne peuvent constituer à elles seules un support d'apprentissage pour l'enseignant. Les fiches pédagogiques peuvent typiquement être utilisées pour illustrer un cours d'architecture des ordinateurs et des composants électroniques.

Prérequis pour les étudiants

Chacune des cinq fiches indique les prérequis nécessaires à sa compréhension. Les prérequis portent sur les connaissances en cryptographie (notions de confidentialité, d'intégrité et d'authentification), en cycle de conception des composants électroniques, en architecture logicielle et matérielle et en électronique : principe des algorithmes de chiffrement, architecture à base de processeur, accélérateur matériel (par exemple coprocesseur), électronique (par exemple courant, tension, onde électromagnétique). Des connaissances détaillées en cryptographie ne sont pas nécessaires (connaissance précise des algorithmes de cryptographie), mais elles permettent d'approfondir certains concepts présentés dans les différentes fiches.

Prérequis pour les formateurs

Les fiches apportent des repères pédagogiques aux enseignants, en présentant de manière structurée et concise les sujets importants de la sécurité des composants électroniques qui peuvent être présentés à des étudiants durant un cours concernant les architectures des ordinateurs et la conception des composants électroniques logiciels et matériels. Ces fiches ne constituent pas un cours complet sur la sécurité des composants électroniques. Il n'est pas demandé à l'enseignant de parfaitement maîtriser le domaine de la sécurité, mais il devra se renseigner sur les sujets présentés pour pleinement exploiter les fiches pédagogiques. Une bonne connaissance des architectures des ordinateurs et des architectures logicielles et matérielles est conseillée.

Utilisation du guide pédagogique

Ce document contient cinq fiches pédagogiques à destination des enseignants dans le domaine de la conception de systèmes à base de composants électroniques logiciels et matériels dans l'enseignement supérieur. Chaque fiche permettra à l'enseignant d'illustrer son cours avec des notions de sécurité. Typiquement, l'enseignant consacrera une à deux heures à la sécurité à la fin de chacun de ses chapitres. Les fiches peuvent être présentées en tout ou partie, dans l'ordre approprié à l'enseignement et aux étudiants visés. Les fiches 1, 2 et 3 sont toutefois présentées dans ce document en suivant un ordre logique qu'il est recommandé de suivre.

Des liens vers des documents en français ou en anglais sont fournis. Les fiches incluent des références vers des livres, articles et sites web permettant d'approfondir les sujets abordés.

Ci-dessous est proposé un récapitulatif des sujets abordés, ainsi que le temps recommandé pour présenter les sujets et les prérequis correspondants.

Numéro	Sujet	Durée	Prérequis
Fiche 1	Introduction aux attaques par canaux auxiliaires	1 heure	Électronique
Fiche 2	Exemples d'attaques par canaux auxiliaires	2 heures	Cryptographie, architecture des ordinateurs et des composants électroniques
Fiche 3	Attaques par injection de fautes	1 heure	Électronique, cryptographie, architecture des ordinateurs et des composants électroniques
Fiche 4	Primitives de sécurité pour les composants électroniques	2 heures	Architectures matérielles à base de processeurs et de composants FPGA
Fiche 5	Sécurité de la conception et protection de la propriété intellectuelle	1 heure	Cycle de conception

1 Fiche 1 : Introduction aux attaques par canaux auxiliaires

1.1 Thématique

Thématique	Attaques par canaux auxiliaires	**Numéro de fiche**	1	**Mise à jour**	07/03/2016

1.2 Thème des cours visés

Cette fiche peut compléter un cours d'implémentation matérielle dans un cursus d'électroniciens, ou un cours d'architecture des ordinateurs (cache de données et prédiction de branches d'un processeur) pour des informaticiens et électroniciens.

1.3 Volume horaire

Le cours de base comprend 15 minutes de présentation sur les attaques par canaux auxiliaires par rapport aux attaques cryptographiques. 15 minutes seront nécessaires pour illustrer les différents types de fuites matérielles et logicielles. Ensuite, il faudra 15 minutes pour traiter l'exemple de la recherche de mot de passe et enfin 15 minutes sur les contre-mesures.

1.4 Prérequis / corequis

Les étudiants doivent avoir quelques notions de cryptographie et des notions d'électronique sur la consommation du matériel ou sur le fonctionnement des processeurs peuvent être introduits.

1.5 Objectifs pédagogiques

L'objectif de cette activité est de sensibiliser les étudiants à la sécurité des implémentations. Il est possible de mesurer différentes grandeurs physiques lors de l'exécution d'un algorithme, appelées *fuites d'information*, qui vont dépendre des implémentations. Ces informations peuvent alors être exploitées très efficacement afin de retrouver des variables secrètes. Enfin, nous présentons quelques mécanismes permettant de se protéger contre ces attaques.

L'idée de base de ces attaques est l'observation que la puissance consommée dans une carte à puce dépend des instructions et des données manipulées. Par exemple, la multiplication consomme plus de puissance qu'une addition, et l'énergie nécessaire pour mettre un bit à 1 ou à 0 est généralement différente.

1.6 Conseils pratiques

Cette fiche est une introduction aux attaques par canaux auxiliaires. Elle décrit les concepts importants et peut être suffisante pour certains étudiants qui veulent savoir ce que recouvrent ces attaques sans aller dans les détails de certaines attaques qui seront décrites dans la fiche 2.

1.7 Description

Les *implémentations* de mécanismes de sécurité et de cryptographie sont une source de vulnérabilités. Le domaine des attaques par canaux auxiliaires s'est développé depuis les années 1980 afin de sécuriser les implémentations sur carte à puce. Il concerne aujourd'hui aussi bien les implémentations matérielles en ASIC *(Application-Specific Integrated Circuit)* ou FPGA *(Field-Programmable Gate Array)* que les implémentations logicielles exécutées dans un ordinateur ou un smartphone. La spécificité de ces attaques est d'utiliser des informations supplémentaires, comme le temps de calcul, la consommation de puissance et le rayonnement électromagnétique, afin d'obtenir des informations sur les données secrètes manipulées pendant le calcul. Ces informations sur des variables internes ou opérations du calcul peuvent être exploitées afin de retrouver les éléments secrets comme les clés secrètes ou des codes PIN *(Personal Identification Number)*. Enfin, de nombreuses contre-mesures ont été proposées pour diminuer les fuites.

Modèle de sécurité : la boîte grise. Le cours doit d'abord présenter comment ces attaques se différencient des attaques *classiques* en cryptographie. Dans le modèle d'attaque traditionnel (boîte noire), les adversaires interrogent la fonction de chiffrement paramétrée par une clé secrète et tentent de retrouver la clé à l'aide des entrées/sorties d'un algorithme cryptographique. Dans le modèle des attaques *par canaux auxiliaires* (boîte grise), décrit dans la figure 1, l'adversaire obtient en plus des entrées/sorties des informations qui dépendent des données publiques et secrètes, comme la valeur de variables internes du calcul. Comme ce modèle est plus fort, les buts de l'adversaire sont plus variés : rechercher des informations sur l'algorithme utilisé (dans le cas d'un algorithme secret) ou rechercher les clés ou variables secrètes d'un algorithme. Les informations supplémentaires sont obtenues en mesurant différentes grandeurs physiques. En effet, la cryptographie est implantée dans de nombreux objets du quotidien (cartes à puce, passeport électronique, *box* internet) et on peut supposer que l'attaquant a un accès physique ou à distance à ces composants et qu'il peut effectuer différentes mesures.

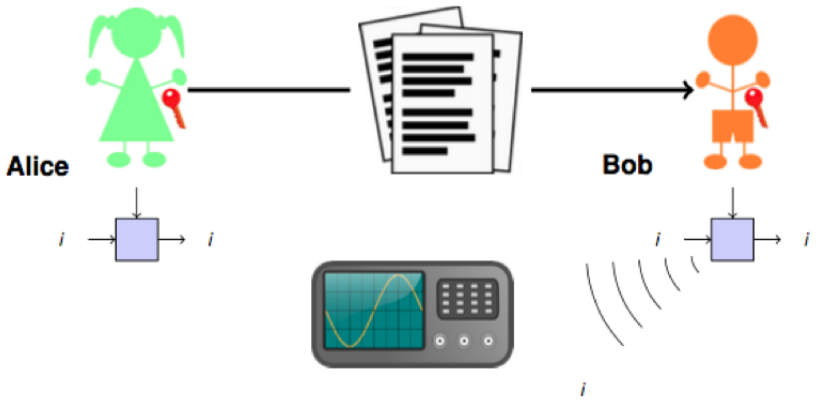

Figure 1 – Attaques par canaux auxiliaires.

Classifications des diverses attaques. Le cours doit ensuite aborder les différents types de *fuites d'information* et les différents types d'attaques. Lors d'une attaque *passive*, l'adversaire observe les exécutions (temps de calcul, courbes de consommation de puissance, rayonnement électromagnétiques). Nous décrirons ces attaques dans cette fiche et la fiche 2. Lors d'une attaque *active*,

l'adversaire injecte des fautes pendant le calcul (fiche 3) en accélérant l'horloge ou en perturbant le rayonnement électromagnétique. Ces perturbations peuvent parfois détruire le composant.

Liens entre les fuites d'information et les algorithmes. L'enseignant doit d'abord montrer les liens entre les fuites et le comportement des algorithmes. Un exemple parlant est celui de la courbe de consommation d'un calcul de l'algorithme de chiffrement AES composé de 10 tours : il est possible de distinguer graphiquement des portions de courbes similaires correspondant à dix passages dans une boucle `for` comme le montre la figure 2 à gauche (en regardant le bas de la courbe qui descend et remonte 10 fois). La dixième itération de la boucle de l'algorithme AES est plus rapide (car l'opération MixColumns n'est pas évaluée dans cette itération). En zoomant sur la portion de la courbe qui correspond à une itération (partie droite de la figure), on peut même isoler les quatre opérations qui composent un tour de la fonction AES. Cette observation peut être la première étape d'une attaque, et illustre le fait que la fuite donne de l'information sur l'algorithme.

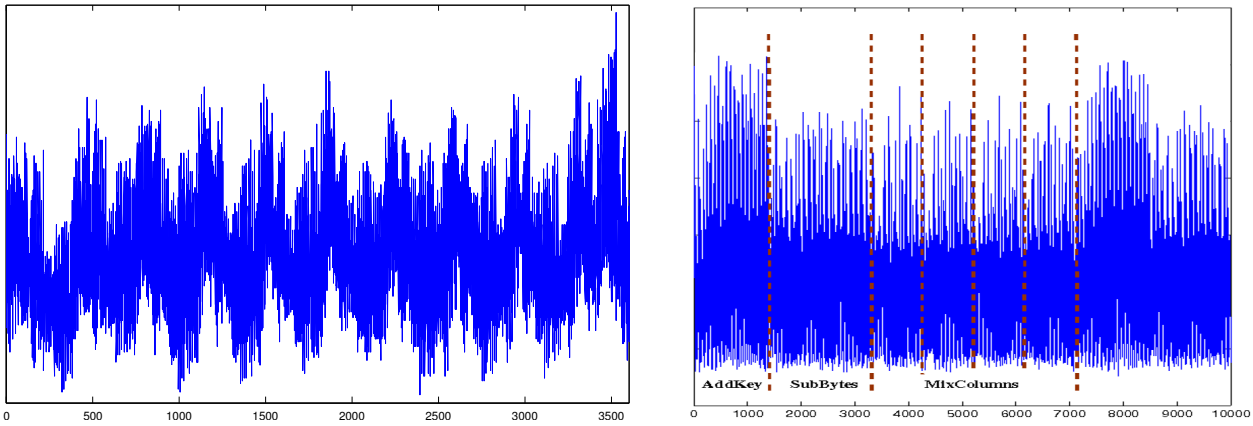

Figure 2 – Figure de gauche : Visualisation des 10 tours de l'algorithme de chiffrement AES – Figure de droite : Visualisation d'un tour d'AES.

Fuites d'information sur les données. Afin d'exploiter plus d'informations pour remonter aux éléments secrets, on peut montrer qu'il existe un lien entre les valeurs manipulées (dépendant du secret) et les fuites dans le cas d'une implémentation matérielle. On peut expliquer par exemple que la puissance consommée dépend des valeurs manipulées dans un composant. En fonction des technologies matérielles utilisées, l'enseignant peut décrire comment le poids de Hamming d'un registre (nombre de bits à 1) et la distance de Hamming (différence des poids de Hamming entre la valeur précédente et la nouvelle valeur), sont liés à la courbe de consommation de courant [3]. La consommation de courant dépend du nombre de bascules qui passent d'une valeur représentant l'état 1 à l'état 0 et dans l'autre sens ; c'est-à-dire de la distance de Hamming. Dans la figure 3, on voit la courbe de consommation en fonction du poids de Hamming d'un registre de 8 bits. Ces résultats montrent comment les données modifient les niveaux de puissance. Ces 9 courbes ont été obtenues en accédant à une donnée par l'instruction LOAD et en moyennant 500 échantillons pour réduire le bruit. La différence de tension entre deux courbes sur la figure est d'environ 13 mV. Il est important de mentionner que le rayonnement électromagnétique permet aussi d'obtenir une information *locale* sur la fuite d'information, alors que la consommation de courant est une information *globale* du circuit.

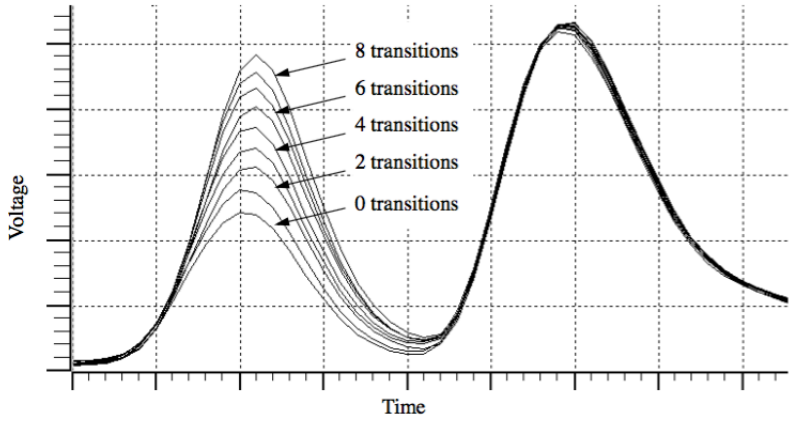

Figure 3 – Courbe de consommation du chargement d'un registre de 8 bits en fonction du poids de Hamming. (Source : [3]).

Fuites d'information sur du logiciel. Le temps de calcul est une variable facile à mesurer sur une implémentation. C'est une information sensible et nous verrons dans la suite comment il est possible d'exploiter cette information globale sur un algorithme pour retrouver un code PIN. De nombreux mécanismes d'architecture des ordinateurs améliorent les performances des processeurs comme le cache de données ou le mécanisme de prédiction de branches. Ils ont des effets sur le temps de calcul et peuvent mener à certaines attaques [9, 6, 8]. On pourra par exemple expliquer comment on peut obtenir les bits de poids forts d'une valeur qu'on lit dans un tableau [6] en utilisant le cache de données. De même, on peut expliquer comment les mécanismes de prédiction et de *pipeliling* permettent d'obtenir des informations sur une suite d'instructions dont l'ordre d'exécution dépend d'une variable secrète [8]. Enfin, dans [9] un exemple d'attaque via un réseau est décrit, ce qui démontre qu'on n'a pas besoin d'avoir un accès direct sur la machine pour monter ce type d'attaque. Plus récemment, le son a été utilisé comme un moyen d'attaque supplémentaire à distance [7] sur un logiciel.

Algorithme 1 Algorithme de vérification de mot de passe.

Paramètres : $P = (P[0], \ldots, P[\ell - 1])$ et $\tilde{P} = (\tilde{P}[0], \ldots, \tilde{P}[\ell' - 1])$
Résultat : "correct" ou "incorrect"
    for $i = 0$ to ℓ-1 **do**
        if $(P[i] \neq \tilde{P}[i])$ **then**
            return "incorrect"
    return "correct"

Principes des attaques : diviser-pour-régner. La cryptanalyse par canaux auxiliaires permet de monter des attaques plus rapides que celles par force brute en retrouvant le secret par petits morceaux. En effet, si l'information porte sur une petite partie de la clé et qu'un test permet de décider si on a fait le bon ou le mauvais choix de la partie recherchée, alors on pourra faire une attaque par force brute sur une partie plus petite ; ce qui accélèrera globalement la recherche du secret. C'est le même principe que pour jouer au *Mastermind*, car on va exploiter des informations partielles sur la solution afin d'accélérer la recherche. Pour illustrer ce type d'attaque, l'exemple le plus simple est une attaque sur un algorithme de vérification de mot de passe qui vérifie caractère par caractère le mot de passe. Une fonction comme `strcmp` aura exactement le même comportement que l'algorithme 1. Le temps de calcul va dépendre du nombre de comparaisons et l'information du

temps passé *au total dans les comparaisons de caractères* permet d'apprendre combien de caractères sont corrects (comme expliqué dans [5]). Si l'enseignant veut monter un TP, il pourra obtenir le temps de calcul grâce à l'instruction `RDTSC` sur les processeurs x86.

Description de l'attaque. Le principe de l'attaque est dans ce cas très simple et utilise des algorithmes de type *diviser pour régner* avec des mécanismes de recherche exhaustive. La recherche exhaustive du mot de passe demande un temps de calcul exponentiel en la longueur de ce dernier. Cette attaque, aussi appelée *force brute*, essaie tous les mots de passe composés de ℓ caractères où chaque caractère est choisi uniformément dans un alphabet A de taille N. Elle demande N^ℓ tentatives. Si $N = 64$ et que la taille du mot de passe est d'au plus 10 caractères, le nombre de mots de passe à tester est très grand, 2^{60} (en général, la distribution n'est pas uniforme, ce qui diminue fortement le nombre d'essais). Si on est capable de mesurer le temps de calcul de façon précise, on peut rechercher le premier caractère en envoyant différents mots de passe qui commencent par toutes les lettres de l'alphabet A, et celui qui aura le temps de vérification le plus long commencera par la même première lettre que celle du mot de passe. Une fois le premier caractère retrouvé, on passe au suivant et ainsi de suite. Par cette méthode, on peut voir que le temps de calcul est $\ell \times N$, ce qui donne 640 essais. Si on veut améliorer la fiabilité des informations (car vérifier une lettre est très rapide), on peut envoyer un ensemble de mots de passe qui ont tous le même premier caractère ou plusieurs fois le même le mot de test, cela augmentera le temps de vérification.

Contre-mesures. Il est important d'expliquer comment se protéger de ces attaques en décrivant les principes de bases de la programmation en temps constant (le temps de calcul ne dépend alors pas des données sensibles) et faire en sorte que l'algorithme effectue toujours les mêmes opérations indépendamment des bits de clés ou de valeurs dépendant du secret [10]. Il existe d'autres mécanismes plus subtils comme le masquage qui consiste à partager un secret en plusieurs morceaux. Ceci oblige l'attaquant à obtenir de l'information sur plusieurs variables s'il veut retrouver le secret et comme les informations sur chaque variable sont bruitées, il devient beaucoup plus difficile de retrouver le secret.

Pour les électroniciens, il sera utile de présenter les contre-mesures matérielles comme les logiques *dual-rail plus pre-charge* (remise à zéro des bus) qui ont une consommation indépendante des données. L'idée de la logique dual-rail consiste à coder un état `0` par deux fils $(0, 1)$ et l'état `1` par $(1, 0)$. De cette manière, il est facile de voir que la consommation sera égale lors des bascules. Il faut mettre en garde les étudiants sur le fait qu'il est *particulièrement* difficile de réaliser en pratique ce type de logique car il est délicat de réaliser un équilibrage parfait entre les deux fils.

D'autres mécanismes permettent aussi de rendre les attaques plus compliquées en exécutant dans un ordre aléatoire les instructions qui peuvent l'être, ou de les désynchroniser dans le temps en introduisant des délais aléatoires, ou enfin de faire tourner en même temps un calcul intensif dont la puissance consommée va noyer la consommation du calcul cryptographique. L'efficacité de ces contre-mesures est à évaluer pour chaque implémentation [11].

1.8 Matériels didactiques et références bibliographiques

[1] ANDERSON Ross, KUHN Markus, *Low Cost Attacks on Tamper Resistant Devices*, 1re édition, 1997. `http://www.cl.cam.ac.uk/~mgk25/tamper2.pdf`

[2] ANDERSON Ross, KUHN Markus, *Tamper Resistance – a Cautionary Note*, 1re édition, 1997. http://www.cl.cam.ac.uk/~mgk25/tamper.pdf

[3] MESSERGES Thomas, DABBISH Ezzy et SLOAN Robert, *Investigations of Power Analysis Attacks Systems*, USENIX Workshop on Smartcart Technology, 1999.

[4] KOCHER Paul, JAFFE Joshua et JUN Benjamin, *Differential Power Analysis*, CRYPTO, 1999. http://www.rambus.com/differential-power-analysis/

[5] JOYE Marc, *Basics of Side-Channel Analysis*, Cryptographic Engineering, Chapitre 13, 2009. http://cs.ucsb.edu/~koc/cren/docs/w05/ch13.pdf

[6] CANTEAUT Anne, LAURADOUX Cédric et SEZNEC André, *Understanding cache attacks*, https://hal.inria.fr/inria-00071387/document

[7] GENKIN Daniel, SHAMIR Adi et TROMER Eran, *RSA Key Extraction via Low-Bandwidth Acoustic Cryptanalysis*, http://eprint.iacr.org/2013/857.pdf

[8] JEAN Jérémy, *Mise en œuvre d'attaques de cryptanalyse basées sur les unités de prédiction de branches*, http://www.di.ens.fr/~jean/pub/ter2008.pdf

[9] BRUMLEY David et BONEH Dan, *Remote Timing Attacks are Practical*, http://crypto.stanford.edu/~dabo/pubs/papers/ssl-timing.pdf

[10] WITTEMAN Marc, *Secure Application Programming in the Presence of Side Channel Attacks*, https://www.riscure.com/benzine/documents/Paper_Side_Channel_Patterns.pdf

[11] CORON Jean-Sébastien et KIZHVATOV Ilya, *Analysis and Improvement of the Random Delay Countermeasure of CHES 2009*, http://www.jscoron.fr/publications/randomdelays2.pdf

2 Fiche 2 : Exemples d'attaques par canaux auxiliaires

2.1 Thématique

Thématique	Attaques par canaux auxiliaires	**Numéro de fiche**	2	**Mise à jour**	07/03/2016

2.2 Thème des cours visés

Cette fiche peut compléter un cours d'implémentation matérielle dans un cursus d'électroniciens.

2.3 Volume horaire

La durée prévue pour cette activité est de 2h. La partie essentielle est composée des attaques simples (SPA) et différentielles (DPA) sur l'algorithme d'exponentiation binaire décrit dans l'algorithme 2 et sur un algorithme de chiffrement symétrique. Il faut compter 30 minutes pour présenter le principe des attaques SPA sur une implémentation de l'algorithme d'exponentiation rapide. Ensuite, 30 minutes seront nécessaires pour présenter le fonctionnement d'un algorithme de chiffrement symétrique, puis 30 minutes pour la DPA et 30 minutes pour les attaques par corrélation (CPA).

2.4 Prérequis / corequis

Les étudiants doivent avoir quelques notions de cryptographie et des notions d'électronique sur la consommation du matériel. La fiche précédente devra être présentée avant cette fiche.

Les principes d'un algorithme de chiffrement symétrique ne sont pas nécessaires, même si cela peut être un plus pour mieux comprendre certains TP. De même, certaines attaques sur RSA ne demandent pas de comprendre RSA mais uniquement le principe de l'algorithme d'exponentiation rapide et l'arithmétique modulaire. Enfin, il peut être utile de connaître quelques éléments de statistique et de compléter si besoin la présentation de la partie sur les attaques DPA par quelques éléments de test du coefficient de corrélation pour les attaques [4] si le TP porte sur le *DPA contest*.

2.5 Objectifs pédagogiques

L'objectif de cette activité est de sensibiliser les étudiants à la sécurité des implémentations et au fonctionnement de ces attaques qui permettent de retrouver très rapidement des éléments secrets. Le but de ce cours est d'expliquer comment il est possible d'exploiter les informations (obtenues grâce aux méthodes de la fiche précédente) afin de retrouver des éléments sensibles.

2.6 Conseils pratiques

Pour compléter ce cours, un TP peut être organisé en utilisant les courbes du *DPA contest* organisé par Telecom ParisTech (`http://www.dpacontest.org/v2/`). Il s'agit de mettre en œuvre une attaque par corrélation contre AES sur le dernier tour. Il existe une implémentation de l'attaque de référence.

2.7 Description

Cette fiche présente différents types d'attaques par canaux auxiliaires. On s'intéresse aux attaques *passives* qui mesurent certaines grandeurs physiques (consommation de courant) et qui ont accès à des entrées ou sorties des algorithmes. On commence par décrire les attaques les plus simples nécessitant une seule courbe d'observation pour retrouver la clé secrète, appelées *Simple Power Analysis* (SPA). Puis, on présente les attaques différentielles (DPA) qui utilisent des techniques statistiques très efficaces. L'idée des attaques physiques est d'exploiter des informations soit sur les *opérations*, soit sur les *opérandes*.

Description de l'exponentiation modulaire. On peut prendre l'exemple d'un calcul RSA manipulant un secret d et calculant une exponentiation modulaire $M^d \bmod N$ à partir d'un message M et du module N pendant une opération de signature ou de déchiffrement [1]. Si on est capable de retrouver d, alors la sécurité de RSA s'effondre. L'algorithme d'exponentiation modulaire est rappelé dans l'algorithme 2. Il utilise la décomposition binaire de $d = \sum_{i=0}^{n-1} d_i 2^i$. Un rappel sur le fonctionnement de l'algorithme peut être fait, mais la compréhension du fonctionnement de l'algorithme n'est pas nécessaire pour comprendre l'attaque. L'algorithme calcule $C = M^d \bmod N = \prod_{i=0}^{n-1} M^{2^i d_i} \bmod N$.

Algorithme 2 Algorithme de d'exponentiation binaire.

Paramètres : M, N et $d = d_0 \dots d_{n-1}$
Résultat : $M^d \bmod N$

```
X = 1
for i = n − 1 to 0 do
    if (d_i = 1) then
        X = X × M mod N        // multiplication
    X = X^2 mod N              // élévation au carré
return X
```

Dans la fiche 1, nous avons expliqué qu'il existe des liens entre les observations du courant électrique par exemple et les algorithmes en observant la courbe de consommation d'un calcul de chiffrement AES. De même, la courbe de consommation de l'algorithme 2 permet également d'observer des répétitions de motifs pour cette boucle `for` (voir figure 4).

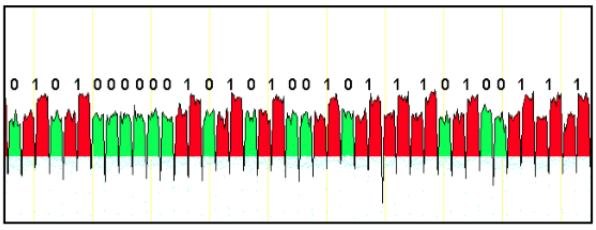

Figure 4 – Courbe de consommation d'un calcul d'exponentiation binaire.

1. En pratique, ce n'est pas le message que l'on signe directement, mais son haché. On parle en effet du paradigme *hash and sign*.

Attaque simple par analyse de courant. Les attaques SPA exploitent des information sur les opérations : si la suite des instructions effectuées dépend des bits de clés et qu'on peut distinguer les opérations (la consommation d'une élévation au carré est moins importante que celle d'une multiplication), alors il est possible de retrouver la clé. C'est le principe des attaques SPA contre l'algorithme RSA : à l'aide du code de l'algorithme 2 qui décrit l'exponentiation binaire, il est possible de voir qu'on peut obtenir les bits de la clé secrète en distinguant les opérations d'élévations au carré, des opérations de multiplications. En effet, un bit à 1 sera composé d'une multiplication suivie d'une élévation au carré, alors qu'un bit à 0 ne sera composé que d'une élévation au carré (figure 4).

Attaques différentielles par analyse de courant. L'analyse simple du courant exploite des différences de consommation visibles sur une ou quelques traces de courant, chacune pouvant être produite par moyenne de plusieurs exécutions (avec les mêmes données d'entrées) afin de diminuer le bruit de mesure. C'est de la comparaison de ces courbes qu'est obtenue l'information sur le secret. Les analyses statistiques de courant (DPA) effectuent un traitement sur un grand nombre de traces obtenues en faisant varier le message en entrées de l'algorithme. Ce traitement statistique permet de réaliser un test d'hypothèses sur une petite portion de la clé, 8 bits en général (comme dans l'attaque contre le mot de passe fiche 1). Ce test d'hypothèses s'appuie sur la consommation de courant produite par la manipulation d'une donnée intermédiaire de l'algorithme qui ne dépend que du message d'entrée (ou du chiffré de sortie) et de la valeur de la portion de clé, appelée sous-clé.

Attaque DPA. Historiquement, la première de ces méthodes a été introduite dès 1998 par Kocher, Jaffe et Jun [1] dans le monde académique, puis formalisée par Messerges et al. [3] et s'appelle l'analyse différentielle du courant (en anglais, *Differential Power Analysis*, DPA). Elle considère un bit arbitraire d'une valeur intermédiaire de l'algorithme qui ne dépend que du message d'entrée et de la valeur d'une sous-clé. Un tel bit est appelé *bit de sélection*. Pour chaque supposition sur la valeur de la sous-clé, il est possible d'effectuer une partition des traces de courant en deux ensembles : l'un des ensembles contient les traces pour lesquelles la valeur du bit de sélection vaut 0, et l'autre contient les traces pour lesquelles il vaut 1. Il est important de souligner que la valeur calculée pour le bit de sélection dépend à la fois du message d'entrée et de la supposition faite sur la valeur de la sous-clé. C'est une prédiction de la valeur réelle de ce bit. Cette prédiction est systématiquement correcte pour la bonne supposition de la sous-clé, et sera correcte par chance une fois sur deux lorsque la valeur supposée de la sous-clé n'est pas la bonne. Une fois l'ensemble des courbes de courant ainsi partitionné en deux groupes, on calcule la courbe de courant moyenne de chacun des groupes, et on les soustrait l'une à l'autre. Il en résulte une courbe de DPA associée à chacune des suppositions sur la sous-clé. Si la valeur supposée de la sous-clé est correcte, alors le bit de sélection reflète exactement sa valeur dans la carte au moment de l'exécution et la courbe de DPA montre une différence de courant significative à chaque instant où ce bit a été manipulé. Cela se traduit par un *pic* à chacun de ces instants. En revanche, si la supposition est incorrecte, les courbes de chaque paquet sont indépendantes de ce qui s'est réellement passé dans la carte pour ce bit, et les courbes de DPA qui en résultent vont être plates au bruit près. La figure 5 présente à mi-hauteur une courbe de DPA sur le début d'un DES qui montre cinq pics prononcés pour la bonne supposition de sous-clé. Dans cette figure, on a représenté la courbe de DPA et on voit bien qu'il y a plusieurs instants où est manipulée le bit de sélection.

Il est ainsi possible d'identifier la bonne valeur de la sous-clé comme étant celle qui produit le pic le plus haut parmi toutes les courbes de DPA. En procédant de la sorte successivement pour toutes les sous-clés, on retrouve progressivement la majeure partie ou la totalité de la clé. Un avantage

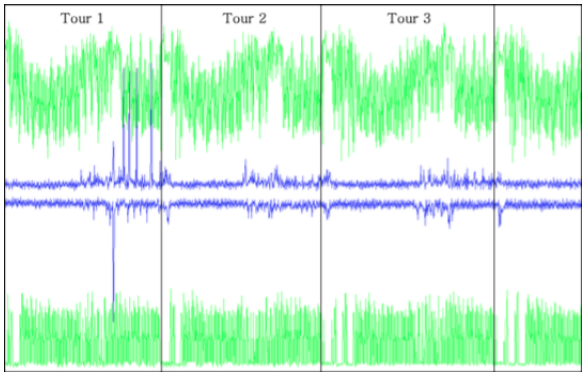

Figure 5 – Courbe DPA d'un calcul DES.

pratique de la DPA est de ne pas nécessiter d'hypothèse forte sur le modèle de consommation en fonction des données. Il est simplement suffisant que la consommation de courant, lorsqu'un mot de donnée est manipulé, soit différente en moyenne selon que l'un des bits de ce mot vaille systématiquement 0 ou systématiquement 1. En revanche la DPA présente l'inconvénient d'être sujette au problème dit des pics fantômes. Contrairement à ce que voudrait la théorie présentée ci-dessus, il peut arriver que certains pics soient présents sur des courbes de DPA correspondant à certaines suppositions incorrectes de la sous-clé. Ces pics fantômes, dont l'amplitude peut tout à fait rivaliser avec les *vrais* pics (ceux apparaissant sur le courbe de DPA pour la bonne sous-clé), constituent bien évidemment un problème pour l'identification correcte de la valeur de la sous-clé.

Attaque CPA. Pour résoudre ce problème de pics fantômes, une méthode a été proposée par Brier, Clavier et Olivier [4] et appelée analyse du courant par corrélation (en anglais, *Correlation Power Analysis*, CPA). Elle suppose que l'attaquant a préalablement identifié un modèle de consommation et repose sur deux idées essentielles. La première est que, connaissant le message d'entrée et une valeur supposée de la sous-clé, l'attaquant dispose de toute l'information nécessaire pour prédire l'intégralité de la valeur du mot machine pertinent pour l'attaque. La deuxième est que, disposant d'un modèle de consommation, l'attaquant peut transformer sa prédiction de la donnée manipulée en prédiction de la consommation occasionnée par la manipulation de cette donnée. Il est donc possible, pour chaque supposition de la sous-clé, d'établir une série de prédictions (une pour chaque courbe acquise) de la consommation de courant à l'instant pertinent pour l'attaque. Au bruit de mesure et à une éventuelle imprécision de modèle près, cette série reflète les consommations mesurées dans le cas où la supposition sur la sous-clé est correcte. Un calcul du cœfficient de corrélation linéaire entre les consommations mesurées l_i et les consommations prédites $m_i^{s^*}$ révèlera alors facilement l'exactitude de la supposition sur la sous-clé s^* (voir figure 6). En effet, si une erreur est commise sur cette sous-clé, le fait d'avoir tenu compte de tous les bits du mot machine pour fonder la prédiction de la consommation rend alors beaucoup plus improbable que dans le cas de la DPA l'émergence d'un pic fantôme de corrélation.

Il existe deux modèles classiques de consommation utiles pour la CPA utilisant le poids de Hamming ou la distance de Hamming. Ces deux modèles sont basés sur le fait que les plus grandes variations de la consommation de courant en fonction d'une donnée se produisent lorsque cette donnée est présentée sur le bus externe de la CPU. À cet instant, chaque ligne de bus pour laquelle se produit une transition de 0 à 1 ou de 1 à 0 consomme une certaine énergie nécessaire pour réaliser ce basculement d'état. En supposant que les contributions des bits sont additives, et que l'énergie requise est la même quel que soit le bit considéré et quel que soit le sens de la transition, on obtient

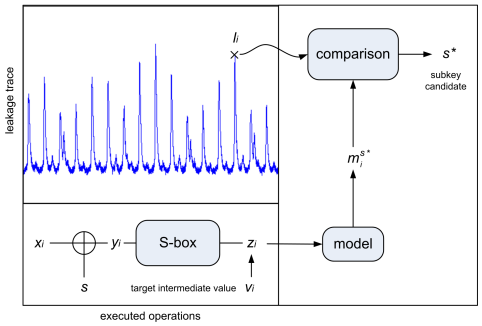

Figure 6 – Modèle d'attaque DPA.

une consommation globale qui est une fonction linéaire de la distance de Hamming entre la valeur de la donnée considérée et celle préalablement présente sur le bus et appelée état de référence. Sur certains composants, le bus est dit préchargé, c'est-à-dire que l'état de référence est remis à zéro (ou à son complément) avant chaque écriture. Dans ce cas de figure, l'état de référence ne joue aucun rôle et le modèle de consommation se réduit à une fonction linéaire en le poids de Hamming de la donnée manipulée. Ces deux modèles de consommation sont les plus couramment adoptés car ils s'ajustent souvent assez bien aux consommations effectivement mesurées. Cette attaque est décrite en détail dans le chapitre 3 page 37 de [5].

Pour aller plus loin. Pour éviter les attaques différentielles, on peut utiliser des techniques de masquage [2]. Enfin, l'enseignant peut mentionner les attaques d'ordres élevées qui combinent plusieurs points de fuite.

2.8 Matériels didactiques et références bibliographiques

[1] Kocher Paul, Jaffe Joshua et Jun Benjamin, *Differential Power Analysis*, CRYPTO, 1999. `http://www.rambus.com/differential-power-analysis/`

[2] Chari Suresh, Jutla Charanjit, Rao Josyula et Rohatgi Pankaj, *Towards Sound Approaches to Counteract Power-Analysis Attacks*, CRYPTO, 1999. `http://www.cs.dartmouth.edu/~ccpalmer/classes/cs55/Content/Resources/dpa.pdf`

[3] Messerges Thomas, Dabbish Ezzy et Sloan Robert, *Power Analysis Attacks of Modular Exponentiations in Smartcards*, `http://saluc.engr.uconn.edu/refs/sidechannel/messerges99power.pdf`

[4] Brier Eric, Clavier Christophe et Olivier Francis, *Correlation Power Analysis with a Leakage Model*, `http://eprint.iacr.org/2003/152.pdf`

[5] Clavier Christophe, *Thèse : De la sécurité physique des crypto-systèmes embarqués*, `http://www.di.ens.fr/~fouque/phd-Clavier-2007.pdf`

3 Fiche 3 : Attaques par injection de fautes

3.1 Thématique

Thématique	Attaques physiques	**Numéro de fiche**	3	**Mise à jour**	03/03/2016

Parmi les différents types d'attaques *physiques*, celles par *observation* (voir les fiches 1 et 2), aussi qualifiées d'attaques *passives* ou par *canaux cachés/auxiliaires (side channel attacks)*, ne sont pas le seul type de menaces contre les composants électroniques utilisés dans des dispositifs de sécurité embarqués, comme les cartes à puces, les téléphones et ordinateurs portables, les dispositifs biomédicaux, les systèmes électroniques et informatiques dans les moyens de transport, etc. Les attaques par *perturbation*, appelées encore attaques par *injection de fautes* ou attaques *actives*, sont une source de vulnérabilités importante qu'il faut prendre en compte dès la conception de composants électroniques de sécurité ou de code embarqué (qui s'exécute sur un processeur logé dans un circuit intégré).

Cette activité est une *introduction* aux attaques par injection de fautes et aux contre-mesures associées. Elle passera en revue les principaux types de fautes, quelques techniques d'injection correspondantes, puis comment ces fautes peuvent être exploitées pour concevoir des attaques, et enfin les principaux types de contre-mesures. Pour chaque élément, elle présentera les principales notions ainsi que des exemples pour illustrer le propos.

3.2 Thème des cours visés

Cette activité s'inscrit typiquement dans un cours d'architecture des ordinateurs ou de conception de circuits numériques ou de systèmes intégrés sur puce dans un cursus électronique. Dans un cursus informatique, elle peut être envisagée en ne détaillant pas les aspects électroniques les plus poussés (beaucoup des principes évoqués s'appliquent directement à des processeurs exécutant du code embarqué).

3.3 Volume horaire

Le volume total envisagé pour cette activité est d'une heure, qui peut être utilisée comme suit :
- 15 minutes sur la notion de fautes et la présentation de quelques principes et techniques d'injections de fautes ;
- 30 minutes sur des exemples d'attaques ;
- 15 minutes sur des exemples de contre-mesures.

3.4 Prérequis / corequis

Il faudra connaître les attaques par observation comme celles décrites dans les fiches 1 et 2.

Des connaissances de base sur les algorithmes de cryptographie symétrique ou asymétrique peuvent permettre d'illustrer le fonctionnement de certaines attaques dans des cas précis. En particulier, cette fiche utilise l'algorithme RSA comme support d'illustration, toutefois comprendre son fonctionnement mathématique n'est pas nécessaire. Il suffit d'admettre le bien fondé de la séquence de calculs effectués durant l'exponentiation modulaire.

Le vocabulaire et certains concepts dans le domaine de l'injection de fautes sont fortement liés au monde du test de composants électroniques. Il peut être bénéfique de positionner cette activité après un cours de base sur le test dans un cursus électronique (cela permet de passer plus rapidement sur la partie concernant les fautes et de consacrer plus de temps aux attaques).

L'injection de fautes est utilisée dans certaines techniques de rétro-conception, ceci peut être une autre activité liée à la sécurité des composants et à prendre en compte selon le public visé (en particulier pour des concepteurs de circuits).

La possibilité de générer des fautes dans un composant électronique est liée à des événements physiques (p. ex. violation de contraintes temporelles, transfert d'énergie), mais aussi à des forces ou faiblesses de certaines techniques et outils de conception de circuits (logique synchrone/asynchrone, analyse statique de temps, codage des états, techniques de placement et de routage, isolation physique, etc.). Il peut être intéressant pour des étudiants d'un cursus électronique d'étudier certains de ces aspects.

3.5 Objectifs pédagogiques

L'objectif de cette activité est de *sensibiliser* les étudiants aux problématiques des attaques élémentaires basées sur des injections de fautes et aux possibles contre-mesures. Les étudiants doivent pouvoir éviter de développer de futures conceptions (en matériel ou en logiciel embarqué) risquant d'être facilement compromises par des attaques par injection de fautes. Pour chaque point abordé, l'enseignant pourra donner à la fois les grands principes et des exemples pratiques.

3.6 Conseils pratiques

Les principes des attaques par injection de fautes sont assez simples, mais le fonctionnement détaillé de certaines attaques historiques ou issues de la littérature scientifique fait intervenir des algorithmes cryptographiques ainsi que des notions mathématiques et électroniques plus ou moins avancées. Il faudra veiller à utiliser des notions qui seront accessibles au public visé.

La notion de coût, ou de faisabilité pratique d'une technique d'attaque par injection de fautes est assez subjective selon le type d'attaquant considéré (budget, niveau de connaissance, équipements disponibles, disponibilité de multiples composants attaqués en cas d'attaques destructives, etc.). Faire comprendre les différences (parfois importantes) de précision, spatiale et temporelle, entre différentes techniques d'injections peut aider à mieux appréhender la difficulté de réaliser une attaque.

Cette fiche présente un éventail de techniques et d'exemples. Il faudra en sélectionner certains selon les connaissances et les intérêts du public visé (p. ex. ne décrire que 2 ou 3 techniques d'injection et mentionner les autres, détailler un seul exemple d'attaque et faire comprendre le principe des autres).

3.7 Description

Les attaques par injection de fautes sont utilisées pour *perturber* le composant électronique afin qu'*il se comporte de façon différente* par rapport à sa spécification, ou pour obtenir des informations sur les valeurs secrètes manipulées dans le circuit (article d'introduction de 13 pages [1], livre dédié à ce sujet [2]). Dans le premier cas, il s'agit d'*invalider des mécanismes de sécurité* (p. ex. ne pas effectuer un test, modifier l'état courant d'une machine à état, etc.), en empêchant l'exécution

d'une instruction ou en modifiant la valeur d'une mémoire ou d'un registre. Dans le deuxième cas, le principe est un peu différent des attaques par observation (voir les fiches 1 et 2). L'occurrence d'une faute va provoquer un changement de comportement d'un élément du composant, qui sera *mesurable à l'extérieur du composant* (plus ou moins directement). Si ce changement est lié à la fois à la faute et à l'information secrète cible, alors l'attaque par injection de fautes permettra d'obtenir des informations là où une attaque par observation ne serait pas suffisante. Il faut faire comprendre que le « à la fois » est important. Il faut en effet que l'occurrence de la faute puisse permettre de « remonter » jusqu'à une information secrète à partir de la manifestation mesurable à l'extérieur du circuit de la faute (ou de son effet sur le fonctionnement). La difficulté d'exploiter une faute (ou des fautes) pour mener à bien une attaque est liée à la complexité du « chemin » pour effectuer cette « remonéer ».

Les fautes génèrent des altérations de caractéristiques physiques ou temporelles d'une portion, plus ou moins importante, d'un circuit électronique. Ces altérations peuvent être permanentes (le composant reste altéré après l'injection de la faute) ou bien temporaires ou transitoires (pour des durées plus ou moins longues selon les cas).

Différentes caractéristiques, paramètres physiques ou temporels peuvent être utilisés pour mener à bien des attaques par injection de fautes :

- Perturbation de la *température* : les circuits intégrés ont des plages de fonctionnement garanties pour chaque technologie. En dehors de ces plages, certaines opérations peuvent être altérées [3] (p. ex. écriture possible mais lecture impossible dans des mémoires non volatiles, modifications aléatoires de données dans des RAM, etc.).
- Perturbation de la *tension d'alimentation* : la distribution des alimentations d'un circuit intégré est faite pour que tous les éléments fonctionnent correctement (les valeurs des états logiques sont codés par des niveaux de tension qui sont reconnus comme valides uniquement dans certaines plages comme en logique CMOS, les délais de propagation sont suffisamment rapides par rapport à la période choisie pour l'horloge du circuit). En faisant chuter brutalement la tension d'alimentation pendant un bref instant, on peut empêcher une bascule de mémoriser une nouvelle donnée, on peut ralentir un signal qui ne sera pas stable au prochain front d'horloge, etc. Au contraire, en augmentant un peu la tension d'alimentation temporairement, on peut accélérer des éléments d'un circuit (p. ex. forcer l'instruction $n+1$ à se finir avant l'instruction n dans un pipeline de processeur). Voir par exemple [4].
- Perturbation des *signaux d'horloge* : les circuits synchrones sont basés sur le fait que dans le pire des cas, le front d'horloge arrive toujours après que l'élément le plus lent du circuit se soit stabilisé. En raccourcissant certains états pendant un ou plusieurs cycles d'horloge, on peut empêcher la mémorisation de certaines valeurs. Cela peut engendrer des problèmes de décodage d'instruction dans un processeur ou un micro-contrôleur [5].
- Perturbation *électromagnétique* : les courants induits par le rayonnement électromagnétique s'ajoutent aux courants déjà présents dans certains éléments du circuit. Ces courants peuvent modifier des valeurs logiques ou bien modifier la vitesse de propagation de signaux (ou la fréquence interne de certains éléments). Voir par exemple [6].
- Perturbation *optique* (par lumière blanche ou tir laser) [7] : les circuits électroniques sont sensibles à l'effet photoélectrique. Les photons induisent des courants électriques additionnels qui peuvent altérer le fonctionnement local (tir laser) ou global (lumière blanche) du circuit. Une préparation physique du circuit est parfois nécessaire (p. ex. décapsulation, polissage de la couche de passivation, amincissement du substrat) pour réussir ce type d'injection. Dans ce cas, on parle d'attaque semi-invasive (contrairement aux attaques non-invasives qui ne modifient en rien le boîtier du circuit).

– Perturbation par *rayonnement ionisant* : certains rayonnements se propagent assez profondément dans la matière et peuvent induire des altérations similaires aux perturbations optiques.
– Attaques *invasives* : à l'aide d'équipements particuliers, on peut placer des micro-sondes sur certaines pistes du circuit (rendues accessibles par une préparation spécifique) pour espionner ou injecter des signaux. Il existe aussi des procédés pour couper ou ajouter des connexions dans un circuit existant, ou bien pour reconstituer le schéma électrique à partir de photographies des différentes couches du circuit qui est découpé par polissage couche par couche. Ces techniques peuvent parfois mener à la destruction du composant. Ces techniques dépassent le cadre de cette introduction.

Plusieurs modèles de fautes logiques sont utilisés pour caractériser les attaques : *collages (stuck-at)*, où le signal est forcé à 0 ou à 1, *inversions (bit-flip)*, qui change la valeur initiale en son complémentaire. Selon le type de méthode d'injection utilisé, seules certaines fautes logiques sont possibles (p. ex. pour un circuit donné, un tir laser peut provoquer des collages à 0 mais pas à 1, ou bien le contraire). L'article [8] fournit une introduction aux modèles de fautes et à leurs implications dans le domaine du test de circuits. La possibilité d'un *contrôle spatial* plus ou moins précis de la position de la faute peut donner un avantage certain à l'attaquant, mais au prix d'un coût d'attaque de plus en plus important (p. ex. réussir à cibler le décodeur d'instruction du processeur, le bus vers la mémoire, les entrées-sorties d'un bloc de cryptographie). Parallèlement à la position, la maîtrise de la *focalisation*, ou la surface d'action de la faute (un bloc complet de l'architecture, un registre particulier, un bit particulier), est aussi un avantage certain. Les SEU *(single event upset)* sont des inversions d'un seul bit, alors que les MEU *(multiple event upset)* concernent plusieurs bits. Le *contrôle temporel* de l'injection de fautes est aussi très important pour réussir une attaque. Plus le contrôle de l'instant et de la durée d'injection seront précis (injections aléatoires pendant une période temporelle plus ou moins importante, *glitch* affectant plusieurs cycles d'horloge, *glitch* dans un cycle précis), plus l'attaquant sera capable de mettre en faute des éléments du circuit à des instants particulièrement propices (p. ex. changer un code instruction avant son décodage, dévier l'adresse d'un saut vers une routine de sécurité vers une autre routine, modifier un état dans une machine à état). Obtenir de grandes précisions spatiales et temporelles est possible mais très *coûteux*, car elles nécessitent des équipements très complexes nécessitant des compétences pointues pour les utiliser. Enfin, la *reproductibilité* peut être un atout important pour les attaques par injection de fautes. Si il est possible d'injecter séquentiellement plusieurs fautes, l'attaquant sera capable de monter une recherche en faisceau sur différentes combinaisons ou bien de valider des hypothèses statistiques pour préparer des attaques peu précises (et donc plus abordables). C'est alors la répétition des injections avec des caractéristiques légèrement différentes qui va permettre de valider ou invalider des hypothèses statistiques.

L'*exploitation* de la faute, ou des fautes, peut être plus ou moins directe dans le fonctionnement de l'attaque. Dans le cas d'une exploitation *directe*, l'altération d'un paramètre ou la corruption d'une valeur (au sens très large car cela peut être un signal de donnée ou bien un signal de contrôle) est directement le but de l'attaque. Les objectifs peuvent être :
– éviter l'exécution d'un test de sécurité ;
– sauter à une adresse précise intéressante pour exécuter directement une routine qui aurait normalement nécessité une certaine forme de vérification (p. ex. vérification d'identité) ;
– changer un état d'une machine à état (FSM, *Finite State Machine*) ;
– forcer une valeur intermédiaire dans un registre pour modifier le comportement d'un algorithme ;
– modifier une instruction devant être exécutée par une autre plus propice à l'attaquant ;
– etc.

Le pseudo-code simple en figure 7 exécute la primitive critique uniquement si l'utilisateur (identifié par une variable) est autorisé (le résultat de la fonction de test est comparé à une valeur attendue).

Dans le cas contraire, le programme lance une fonction d'erreur dédiée.

1 : **if** autorisation(utilisateur) $\neq$ oui **then** 2 : erreur_utilisateur_non_autorisé 3 : **else** 4 : primitive_critique(...)	1 : **load** R1, @utilisateur 2 : **call** @autorisation 3 : **cmp** cmp R7, cst_oui 4 : **bneq** @erreur_utilisateur_non_autorisé 5 : **call** @primitive_critique

FIGURE 7 – Exemple de pseudo-code (à gauche) et pseudo-assembleur (à droite).

Le pseudo-assembleur de la figure 7 est un exemple de résultat de compilation possible pour un système et un processeur très simples. On suppose que le passage de paramètre d'une fonction se fait via le registre R1 et que celui de la valeur de retour se fait par le registre R7. Plusieurs scénarios d'attaques par injection de fautes sont envisageables, entre autres :

- Injecter une faute lors de l'exécution du branchement conditionnel **bneq** en ligne 4. La faute peut empêcher le bon décodage de l'instruction, ou de son exécution. L'instruction exécutée à la place peut être sans lien avec la sécurité ;
- Injecter une faute lors de la récupération (*fetch*) en mémoire externe de l'instruction de branchement **bneq** en ligne 4 pour la remplacer par une instruction **nop** (p. ex. un tir laser qui colle le registre à 00...00 si **nop** est codé 00...00) ;
- Injecter une faute pour falsifier l'exécution de la comparaison en ligne 3 (possible selon le codage des valeurs R7 et cst_oui, ou du fonctionnement de l'unité arithmétique et logique du processeur) ;
- Injecter une faute sur l'adresse de la fonction appelée en ligne 2, normalement celle de test d'autorisation, pour exécuter une fonction qui retourne une valeur dans R7 propice à l'attaquant.

Selon l'architecture interne du processeur et de son fonctionnement, certains scénarios peuvent être exploités ou pas.

Dans le cas d'une exploitation *indirecte*, la corruption d'une valeur va provoquer un changement de fonctionnement qui sera exploitable pour mesurer ce changement (ou une conséquence de ce dernier) depuis l'extérieur et en déduire, plus ou moins directement, une information sur une donnée secrète manipulée dans le circuit. Plusieurs types de scénarios sont possibles, entre autres :

- Attaques de type *safe-error* (nous détaillons ci-dessous un exemple typique pour RSA) : si des opérations inutiles sont exécutées, alors l'injection d'une faute lors d'une telle opération n'aura pas de conséquence sur les valeurs en sortie. Et si l'opération inutile est liée, plus ou moins directement, avec le secret, on peut donc retrouver des parties de ce secret en injectant des fautes à différents instants.
- Utilisation de *statistiques* : l'injection de plusieurs fautes va permettre d'obtenir un ensemble de valeurs pour différentes exécutions d'un même crypto-système mais avec des valeurs intermédiaires non voulues (p. ex. certaines entrées de *SBOX* dans AES). Cela peut donner des informations à l'attaquant. Les *attaques différentielles* ou DFA (*differencial fault attack*) peuvent être très efficaces en pratique [9]. On peut même les combiner avec des techniques de cryptanalyse théorique pour exploiter des liens mathématiques (p. ex. systèmes d'équations) entre les valeurs en faute, des données secrètes et les résultats observés. Mais cela dépasse le cadre de cette introduction.
- Utilisation *combinée* d'attaques par injection de fautes et par observation. Les fautes peuvent modifier le comportement du circuit de façon que l'observation de canaux cachés (p. ex. puissance consommée, rayonnement électromagnétique, temps de calcul) puisse faire « ressortir » une variation d'un paramètre physique corrélé (directement ou statistiquement) à une information secrète [10].

Utiliser une contre-mesure pour contrer une attaque peut très bien affaiblir le composant vis-à-vis d'un autre type d'attaque. Les attaques de type *safe error* illustrent le cas où une contre-

mesure proposée pour se protéger d'attaques par observation va permettre de réaliser une attaque par perturbation impossible avant l'ajout de la contre-mesure. Par exemple, dans le cas de la protection contre la SPA (voir fiche 2) pour l'algorithme RSA, une contre-mesure proposée est la technique *multiply-and-square-always* illustrée dans l'algorithme 3. D'un point de vue SPA, cette contre-mesure fonctionne car la séquence de calculs est parfaitement uniforme (motifs « carré puis multiplication »). Mais elle ajoute une vulnérabilité vis-à-vis de l'injection de fautes. Si l'attaquant est capable d'injecter une faute lors de l'exécution de l'opération de multiplication en ligne 4, alors il pourra facilement savoir si la variable X_1 a été utilisée ou pas lors de la mise à jour de l'accumulateur en ligne 5 (qui ne s'implante pas par une recopie, mais par une sélection d'adresse de la bonne variable). En effet, dans le cas d'un bit de clé d_i égal à 0, seule la variable X_0 est utilisée durant l'itération i. L'attaquant peut donc discriminer assez facilement les bits de clés à 1 ou à 0 en injectant des fautes « petit à petit » durant l'exponentiation modulaire. Ce type d'attaque est très efficace en pratique car l'attaquant n'a pas besoin d'une grande précision spatiale mais « seulement » de bien contrôler l'instant d'injection.

Algorithme 3 Contre-mesure *multiply-and-square-always* pour RSA contre la SPA.

Paramètres : N le module RSA, $d = (d_{t-1}d_{t-2}\dots d_1 d_0)_2$ l'exposant et M un élément
Résultat : $M^d \bmod N$

$X_0 \leftarrow 1$, $X_1 \leftarrow 1$
for i **from** $t-1$ **downto** 0 **do**
  $X_0 \leftarrow X_0^2 \bmod N$ // *carré*
  $X_1 \leftarrow X_0 \cdot M \bmod N$ // *multiplication*
  $X_0 \leftarrow X_{d_i}$ // *sélection de la bonne variable*
return X_0

Dans l'autre sens, l'utilisation d'une contre-mesure contre des attaques par injection de fautes ne doit pas affaiblir le système vis à vis des attaques par observation. Par exemple, ajouter des calculs supplémentaires pour vérifier qu'il n'y a pas de faute dans les calculs intermédiaires peut constituer une source de fuite d'information lors d'une attaque par canaux auxiliaires.

Pour certains composants électroniques de sécurité, comme des générateurs de nombres vraiment aléatoires TRNG (*true random number generator*), des variantes d'attaques par injection de fautes sont utilisables pour réduire la qualité de l'aléa en sortie du TRNG. Par exemple, dans le cas des TRNG basés sur de multiples oscillateurs en anneaux, une injection électromagnétique avec une fréquence particulière et une énergie suffisante, peut faire que les différents anneaux se verrouillent sur cette fréquence au lieu d'osciller librement à des fréquences différentes. Comme la source d'aléa est basée sur la gigue de phase relative (supposée aléatoire) entre les différents anneaux, l'injection de la fréquence d'attaque va réduire la qualité de l'aléa généré, cela même si aucune faute, au sens logique (collage, *bit-flip*), n'a été injectée dans le circuit.

Dans la littérature scientifique (comme l'article de présentation de l'état de l'art [11]), il existe de nombreuses contre-mesures pour protéger un composant électronique contre des attaques par injections de fautes. Toutefois, quelques grands principes de protection s'appliquent en général :

- Utilisation de *protections matérielles* : certains capteurs intégrés au sein du composant (dans le boîtier, ou directement sur la puce) permettent de détecter des changements de la tension d'alimentation, des variations de fréquence, ou bien des rayonnements, et donc de mettre en œuvre un mécanisme de sécurité. On peut aussi utiliser un blindage électromagnétique autour de la puce qui limitera beaucoup les attaques de ce type.
- Utilisation de *redondances* spatiales et/ou temporelles : Rejouer les calculs (séquentiellement ou en parallèle) permet de comparer plusieurs exécutions. Si les rejeux sont aléatoires, il faudrait à

l'attaquant beaucoup de chance pour réussir à injecter des fautes aux bons moments pour que deux exécutions liées soient également fausses de la même façon. Différents niveaux d'application des redondances sont possibles : porte logique, opération, fonction, bloc. Simple à mettre en œuvre, ce type de protection à souvent un coût important (en temps, en surface et en énergie consommée).
- Utilisation de *propriétés mathématiques/algorithmiques* : On peut vérifier que certaines valeurs sont, régulièrement ou pas, dans un ensemble de valeurs possibles (p. ex. intervalle de définition). La vérification d'invariants au cours des calculs peut aider à détecter certaines erreurs (indices dans des boucles, différence constante entre des valeurs successives dans une séquence d'opérations).
- La *randomisation* des calculs, quand elle est possible, peut aider à protéger le composant contre des attaques par injections de fautes car l'attaquant doit gérer beaucoup plus d'instants d'injection. En particulier pour les attaques différentielles, cela oblige l'attaquant à devoir « synchroniser » ses traces avec de très nombreux calculs. Mais pour ce type de protection, il faut faire attention aux problèmes de temps de calcul variable liés aux attaques par canaux cachés. De plus, comme souvent en cryptographie, la qualité de l'aléa doit être suffisamment bonne.
- La *représentation des données* peut être également un élément important pour se protéger. Par exemple, dans le cas de l'attaque du pseudo-assembleur en figure 7, le codage de l'instruction **nop** par un mot uniquement composé de 0 est dangereux si des collages à 0 sont envisageables dans le processeur. Il faut alors coder les instructions et les données (mais aussi le contrôle des parties critiques) avec des valeurs qui évitent d'être facilement forcées par une injection de fautes (collages). Dans le cas des instructions d'un processeur, cela complique le décodage et pénalise les performances.
- Utilisation de *codes correcteurs d'erreurs* (ou *error code corrector* ECC qui ne doit pas être confondu avec le sigle ECC pour *elliptic curve cryptography*). Ajouter des blocs de détection/-correction d'erreurs en plus des données (au sens large, cela peut inclure des signaux de contrôle) permet souvent d'obtenir des performances de sécurité équivalentes aux techniques de redondances. Mais ces dernières sont complexes à mettre en œuvre (en particulier pour les opérations sur les corps finis ou les anneaux omniprésents dans AES, RSA et les courbes elliptiques).

La notion de politique de sécurité peut être rapidement abordée pour expliquer que le problème est complexe. Des techniques de détection ou bien de tolérance aux fautes peuvent être utilisées. Le caractère actif ou pas de la technique utilisée a un impact important sur la politique de sécurité recherchée (p. ex. doit-on arrêter un chiffrement lors d'une détection ou bien fournir un résultat faux/aléatoire ?).

Le vieillissement naturel des composants électroniques va altérer certaines de leurs caractéristiques physiques ou temporelles. Ceci peut réduire la robustesse du composant à des attaques par injection de fautes (mais aussi par observation) au cours de la vie du composant. De plus, les gravures des circuits étant de plus en plus fines (130 nm, 65 nm, 48 nm, 22 nm, etc.), les niveaux d'énergie mis en œuvre dans chaque transistor sont de plus en plus faibles (il y a moins de charges actives dans chaque élément). Les futures technologies seront donc très probablement de plus en plus sujettes à des fautes naturelles (du fait d'un environnement difficile) ou forcées par des attaques. La robustesse aux attaques par injections de fautes est un élément de sécurité à prendre en compte sérieusement dans l'avenir.

Pour aller plus loin, les chapitres d'introduction et d'état de l'art de thèses du domaine comme [12, 13, 14, 15] constituent un excellent point de départ.

3.8 Matériels didactiques et références bibliographiques

[1] BAR-EL H., CHOUKRI H., NACCACHE D., TUNSTALL M. and WHELAN C., *The Sorcerer's Apprentice Guide to Fault Attacks*, Proceedings of the IEEE, vol. 96, n. 2, pp. 370-382, feb. 2006. DOI : 10.1109/JPROC.2005.862424
https://eprint.iacr.org/2004/100.pdf

[2] JOYE M. and TUNSTALL M., *Fault Analysis in Cryptography*, Springer, 2012. DOI : 10.1007/978-3-642-29656-7

[3] HUTTER M. and SCHMIDT J.-M., *The Temperature Side Channel and Heating Fault Attacks*, Proceedings of CARDIS, LNCS vol. 8419, pp 219-235, 2012. DOI : 10.1007/978-3-319-08302-5_15
https://eprint.iacr.org/2014/190.pdf

[4] KORAK, T. and HOEFLER, M., *On the Effects of Clock and Power Supply Tampering on Two Microcontroller Platforms*, Proceedings of FDTC, pp. 8-17, 2014. DOI : 10.1109/FDTC.2014.11
https://online.tugraz.at/tug_online/voe_main2.getvolltext?pCurrPk=79126

[5] ENDO S., SUGAWARA T., HOMMA, N. AOKI T. and SATOH A., *An On-Chip Glitchy-Clock Generator for Testing Fault Injection Attacks*, Journal of Cryptographic Engineering, vol. 1, n. 4, pp. 265-270, dec. 2011. DOI : 10.1007/s13389-011-0022-y
http://cosade2011.cased.de/files/2011/cosade2011_talk17_paper.pdf

[6] MORO N., DEHBAOUI A., HEYDEMANN K., ROBISSON B. and ENCRENAZ E., *Electromagnetic Fault Injection : Towards a Fault Model on a 32-bit Microcontroller*, Proceedings of FDTC, pp. 77-88, aug. 2013. DOI : 10.1109/FDTC.2013.9 https://hal.archives-ouvertes.fr/emse-00871218/

[7] SKOROBOGATOV S. P. and ANDERSON R. J., *Optical Fault Induction Attacks*, Proceedings of CHES, LNCS vol. 2523, pp 2-12, 2003. DOI : 10.1007/3-540-36400-5_2
http://www.cl.cam.ac.uk/~sps32/ches02-optofault.pdf

[8] LEVEUGLE R., *Test des circuits intégrés numériques - Notions de base. Génération de vecteurs*, Techniques de l'Ingénieur, n. E2460, août 2002.

[9] BIHAM E. and SHAMIR A., *Differential fault analysis of secret key cryptosystems*, Proceedings of CRYPTO, LNCS vol. 1294, pp. 513-525, May 2006. DOI : 10.1007/BFb0052259
http://info.psu.edu.sa/psu/cis/abuelyaman/Research/DFA-Secret-Key.pdf

[10] ROCHE T., LOMNÉ V. and KHALFALLAH K., *Combined Fault and Side-Channel Attack on Protected Implementations of AES*, Proceedings of CARDIS, LNCS vol. 7079, pp. 65-83, 2011. DOI : 10.1007/978-3-642-27257-8_5
http://www.ssi.gouv.fr/uploads/IMG/pdf/DFSCA.pdf

[11] KARAKLAJIC D., SCHMIDT J.-M. and VERBAUWHEDE I., *Hardware Designer's Guide to Fault Attacks*, IEEE Transactions on VLSI Systems, vol. 21, n. 12, pp. 2295-2306, dec. 2013. DOI : 10.1109/TVLSI.2012.2231707
https://securewww.esat.kuleuven.be/cosic/publications/article-2204.pdf

[12] MORO N., *Sécurisation de programmes assembleur face aux attaques visant les processeurs embarqués*, Thèse de l'Université Pierre et Marie Curie, nov. 2014
https://tel.archives-ouvertes.fr/tel-01147122/

[13] MIRBAHA A.-P., *Étude de la vulnérabilité des circuits cryptographiques l'injection de fautes par laser*, Thèse de l'École Nationale Supérieure des Mines de Saint-Etienne, 2011.
https://tel.archives-ouvertes.fr/tel-00844751/

[14] SELMANE N., *Global and local Fault attacks on AES cryptoprocessor : Implementation and Countermeasures*, Thèse de Télécom ParisTech, 2010.
https://tel.archives-ouvertes.fr/pastel-00565881/

[15] ZUSSA L., *Étude des techniques d'injection de fautes par violation de contraintes temporelles permettant la cryptanalyse physique de circuits sécurisés*, Thèse de l'École Nationale Supérieure des Mines de Saint-Etienne, 2014.
https://tel.archives-ouvertes.fr/tel-01134488/

4 Fiche 4 : Primitives de sécurité pour les composants électroniques

4.1 Thématique

Thématique	Protection de la mémoire et du *boot*, protection des communications et protection du *bitstream*	**Numéro de fiche**	4	**Mise à jour**	03/03/2016

L'objectif de cette activité est de sensibiliser les étudiants à la problématique de la protection des données et des ressources au sein des composants électroniques. Les notions de TPM, de protection du *boot*, de protection de la mémoire externe, de protection des communications et de protection du *bitstream* pour les composants FPGA sont abordées. Cette activité correspond à une introduction aux problématiques soulevées tout au long de la fiche et donne plusieurs références bibliographiques afin d'aller plus en profondeur sur certains points suscitant davantage d'intérêt de la part de l'enseignant ou des étudiants.

4.2 Thème des cours visés

- Architecture des ordinateurs
- Composants électroniques

4.3 Volume horaire

Cette activité nécessite deux heures de présentation sous la forme d'un cours. Ces deux heures peuvent se décliner de la façon suivante :
- 15 minutes sur la présentation du modèle de menaces ;
- 15 minutes d'introduction au concept de *Trusted Platform Module* (TPM) ;
- 25 minutes sur la protection du *boot* ;
- 25 minutes sur la protection de la mémoire externe ;
- 25 minutes sur la protection des communications ;
- 15 minutes sur la protection du *bitstream*.

Ce volume horaire peut être étendu si l'enseignant souhaite approfondir certaines solutions de protection. Par exemple, dans le domaine de la protection de la mémoire externe, il existe plusieurs solutions techniques qui peuvent être détaillées (le chapitre 4 de la thèse de Jérémie CRENNE [5] introduit certaines solutions). De la même manière, il est possible de détailler les différentes solutions de protection du *bitstream* selon les fabricants de FPGA (par exemple Xilinx, Microsemi ou Altera). Cette fiche a été construite afin de permettre à l'enseignant d'avoir un panorama des problématiques existantes en termes de primitives de sécurité pour les composants électroniques. L'enseignant peut donc choisir de survoler ces différentes problématiques ou bien de traiter un point plus spécifiquement en l'approfondissant à partir des références bibliographiques données. Les volumes horaires donnés ci-dessus peuvent dont varier selon l'intérêt que porte l'enseignant à tel ou tel point.

4.4 Prérequis / corequis

Cette activité s'intègre dans un cours d'architecture des ordinateurs et des composants électroniques composés de processeurs et d'accélérateurs matériels (par exemple des coprocesseurs). Elle intègre également des notions de protections de *bitstream* pouvant être détaillées dans un cours présentant les composants FPGA. Cette activité peut être introduite dès que les étudiants connaissent les architectures logicielles et matérielles ou les composants FPGA. Elle permet aux étudiants de prendre conscience des problématiques de protections des données et des ressources, et de connaître certaines solutions techniques (contre-mesures). Cette activité peut être présentée sous forme d'une intervention du type cours magistral (CM).

Les solutions de protection des données et des ressources au sein des composants électroniques font appel à des algorithmes de cryptographie (par exemple le chiffrement symétrique, le chiffrement asymétrique et les fonctions de hachage), aussi il peut être intéressant d'associer cette activité à une présentation plus approfondie de certains de ces algorithmes.

4.5 Objectifs pédagogiques

Cette activité vise à sensibiliser les étudiants aux problématiques de protection des données et des ressources, en indiquant que suivant le modèle de menaces considéré, une architecture peut être vulnérable à différents types d'attaques et qu'il est nécessaire d'intégrer des mécanismes de protection.

4.6 Conseils pratiques

Cette activité est présentée sour la forme d'un cours. Une fois que les étudiants auront intégré les concepts architecturaux, il sera important d'indiquer que différentes menaces existent et qu'il est donc important d'ajouter des mécanismes permettant de garantir les propriétés de confidentialité, d'intégrité et d'authentification. Ces concepts clés de la sécurité peuvent être mis en œuvre au sein des systèmes électroniques, aussi, pour chaque point présenté dans cette activité, plusieurs solutions peuvent être déclinées.

4.7 Description

Cette fiche présente tout d'abord les mécanismes de protection au sein des architectures logicielles et matérielles.

Modèle de menaces

Lors de l'exécution d'une application ou d'un système sur un composant électronique un certain nombre de menaces existent [1]. Au niveau matériel, comme expliqué dans les fiches 1 à 3, les menaces peuvent être qualifiées de passives ou d'actives. Ces dernières peuvent intervenir à différent moments, notamment lors de la phase de *boot*, lors de l'exécution des applications ou lors des communications entre les applications. Au moment du *boot* une attaque possible consiste à contourner les tests d'intégrité et d'authentification en injectant, par exemple, des fautes afin de permettre l'exécution d'un code compromis (voir la fiche 3). Lors de l'exécution, une attaque par

canaux auxiliaires peut permettre de récupérer des informations secrètes telles que des clés de chiffrement (voir les fiches 1 et 2). Lors des communications il est possible de modifier les adresses de destination par une attaque en faute afin de détourner les données recherchées (voir la fiche 3). Concernant la protection des données en mémoire externe, plusieurs types de menaces peuvent être considérés : par exemple les données peuvent être modifiées aléatoirement dans la mémoire ou lors du transfert sur le bus d'accès à la mémoire (par exemple en injectant des fautes comme expliqué dans la fiche 3), ou bien d'anciennes données peuvent être rejouées afin d'accéder à un état connu du système (on parle alors de rejeu), ou encore les données peuvent être déplacées dans la mémoire afin d'accèder à un état spécifique du système (dans ce cas les données sont valides mais ne sont pas présentes en mémoire à l'adresse attendue). Ces différentes menaces nécessitent des mécanismes de protection spécifiques qui sont introduits dans cette fiche.

Notion de TPM

Un groupement d'industriel, le TCG (*(Trusted Computing Group)* définit des mécanismes de sécurité depuis une dizaine d'années afin de protéger les architectures logicielles et matérielles. Le TPM *(Trusted Platform Module)* correspond au plus connu de ces mécanismes [2, 4]. Un TPM est un composant cryptographique matériel qui est généralement intégré sur les cartes mères des ordinateurs. Il s'agit d'un composant fonctionnellement esclave : il ne peut pas donner d'ordre à l'ordinateur, bloquer le système ou surveiller l'exécution d'une application. Un TPM est avant tout un composant servant à mémoriser de manière sécurisée des informations secrètes (typiquement des clés de chiffrement), en garantissant la délivrance de ces secrets uniquement dans des conditions maîtrisées. Il peut alors servir de racine de confiance pour le démarrage d'un système.

Le principal cas d'emploi d'un TPM est d'assurer l'intégrité et l'authentification du processus de *boot* d'un ordinateur. Pour cela, le TPM repose sur des registres appelés PCR *(Platform Configuration Registers)*, qui caractérisent l'état de la plateforme. Ces registres sont initialisés à chaque démarrage de la machine à une valeur fixe. Ensuite, chaque étape du démarrage fait évoluer ces registres pour tenir compte de l'état de la machine. On utilise pour cela une opération de *mesure* du code : avant d'exécuter un morceau de code, la valeur d'un des registres est remplacée par le haché de son ancienne valeur, concaténée avec le code à exécuter. Ainsi, à la fin de la séquence de *boot*, la valeur des registres dépend du code exécuté.

Comme le TPM ne permet pas de modifier les registres autrement que par cette opération de mesure, l'état des registres permet de caractériser l'état du code exécuté au démarrage. Il est alors possible de conditionner l'obtention d'une clé de déchiffrement de disque à la vérification de la valeur attendue dans les registres. Si un attaquant modifie le code de *boot*, la clé ne pourra pas être obtenue.

Cette approche a des limites. D'une part, on fait l'hypothèse que les premières instructions exécutées au démarrage ne pourront pas être altérées par un attaquant. De plus, il faut que tous les éléments pertinents pour la sécurité soient *mesurés* : le code du BIOS, le *bootloader*, le noyau du système d'exploitation, mais également les paramètres du BIOS, le code embarqué par les périphériques externes, etc. Enfin, la mise en œuvre du TPM peut se révéler délicate, en particulier dans le contexte d'une mise à jour d'un des éléments mesurés.

Au-delà de ce mécanisme, le TPM propose également d'autres fonctions de sécurité, comme l'attestation à distance de son état interne. Pour cela, le composant signe un ensemble de registres et avec une clé privée embarquée.

Une architecture TPM contient les éléments suivants afin de répondre aux besoins des fonctions de

hachage et de signature :
- une unité de chiffrement/déchiffrement à clé secrète pour garantir la confidentialité ;
- une unité permettant de signer/vérifier des données à clé publique pour garantir l'authentification ;
- une unité de hachage pour garantir l'intégrité ;
- un générateur de nombres aléatoires pour, entre autres, générer des clés ;
- une mémoire EEPROM non volatile pour sauvegarder des secrets ;
- des capteurs pour contrôler, par exemple, la température, la tension d'alimentation, la fréquence de fonctionnement.

Jusqu'à la version 1.2 de la spécification, le TPM devait être un composant physique dédié, dont la structure interne devait comporter un certain nombre de contre-mesures anti-intrusives, de type composant carte-à-puce (notion de coffre-fort électronique). Avec la version 2 de la spécification, le TPM peut être implémentée de manière complètement logicielle.

Différents constructeurs, tels que les entreprises Atmel, Broadcom, Infineon, STMicroelectronics, Toshiba et Intel proposent ainsi des composants intégrant des fonctionnalités de sécurité du type TPM. Les composants OPTIGA^{TM} TPM SLB d'Infineon illustrent bien ce type de solution [3]. En effet, cette architecture contient les différents éléments permettant de gérer des clés, de les sauvegarder de façon sécurisée et de garantir une étape de *boot* sécurisée.

Protection du *boot*

L'étape du *boot* est une étape critique de la mise en route d'un système à base de processeur. En effet, l'étape du *boot* correspond à l'exécution des toutes premières instructions lors de la mise sous tension d'un ordinateur. Ces instructions sont généralement stockées dans une mémoire non volatile (ROM, Flash, EEPROM), et dépendent des spécificités du système considéré. Si un attaquant peut modifier le code du *boot* avant son chargement en mémoire RAM, alors il peut compromettre le contrôle du système. Il est donc essentiel de garantir que le code du *boot* n'a pas été modifié (notions d'intégrité et d'authentification). En revanche, les informations présentes dans le code du *boot* ne sont pas forcément confidentielles. Toutefois, pour renforcer le niveau de sécurité, ce code peut également être chiffré.

Afin de garantir l'intégrité du code du *boot* présent dans la mémoire non volatile, des fonctions de hachage peuvent être mises en œuvre, par exemple les algorithmes SHA-2 ou SHA-3. Sans entrer dans le détail de ces algorithmes, ces derniers permettent d'obtenir une empreinte du code (également appelée condensé ou *tag*). Afin de garantir l'authenticité du code du *boot*, des algorithmes asymétriques peuvent être utilisés. Ces algorithmes permettent, dans ce contexte, de générer une signature en utilisant notamment l'empreinte du code du *boot*. Cette signature est stockée dans la mémoire non volatile. Si au moment de l'étape du *boot* la fonction de vérification de la signature retourne une réponse positive, le code peut être exécuté. En revanche, si une réponse négative est retournée, une exception doit être levée, car cela signifie que le code qui va être exécuté a été modifié (intentionnellement ou non) par rapport au code attendu. La section 4.5 (page 110) de la thèse de Jérémie CRENNE [5] présente une solution permettant de garantir la protection du code du *boot*. La figure 8 illustre le processus de chargement du code du *boot* en faisant apparaître les différentes étapes du chargement du code du *boot* à l'exécution du code.

Comme indiqué précédemment il est également possible de chiffrer le code du *boot* avant de le sauvergarder dans la mémoire non volatile pour garantir sa confidentialité. Dans ce cas, lors de l'étape du *boot* il est nécessaire de le déchiffrer afin de retrouver le code initial. Ce type de technique implique l'utilisation d'une clé secrète qui doit être stockée dans un endroit protégé au sein de l'architecture. Dans ce cas, lors de l'étape du *boot* la signature est tout d'abord vérifiée,

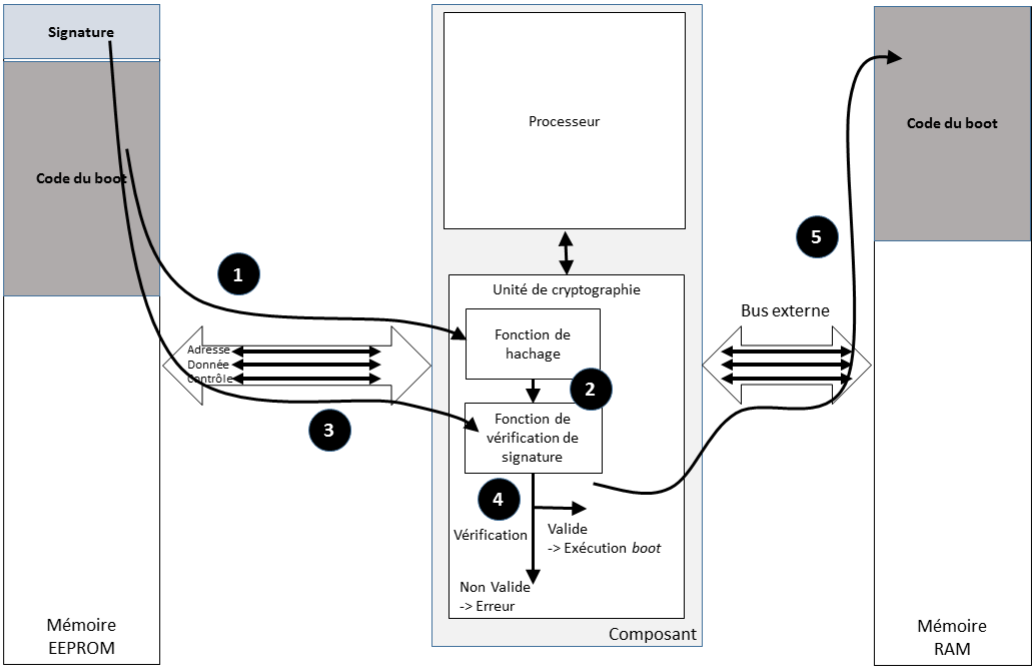

Figure 8 – Protection du boot. Les différentes étapes sont : le chargement du code du *boot* (étape 1), le calcul de l'intégrité (étape 2), le chargement de la signature (étape 3), la vérification de la signature (étape 4), et l'exécution du code (étape 5)

comme indiqué précédemmment, puis si cette dernière est valide, le code est déchiffré et exécuté. Il peut être également nécessaire de traiter la problématique du rejeu, ce point étant important afin de garantir que le code qui va être exécuté correspond bien à la dernière version du code. Il est alors nécessaire d'ajouter une information relative au numéro de version du code dans la mémoire non volatile. Cette information est ensuite utilisée lors de la vérification de la signature au moment de l'étape du *boot*. Par exemple, ce numéro de version sera intégré dans le calcul de l'empreinte avec la fonction de hachage en plus du code lui même.

Protection des mémoires

En ce qui concerne la protection de la mémoire, il existe plusieurs solutions visant à obtenir le meilleur compromis sécurité versus performance en fonction du contexte [5, 6]. Tout d'abord, il est important d'avoir en tête la hiérarchie mémoire, car suivant le niveau d'accès au sein de cette hiérarchie (cache L1, cache L2, mémoire DRAM, disque dur) les contraintes en termes de performance ne sont pas les mêmes. En effet, l'impact temporel est différent s'il s'agit de protéger les données qui sont sur la mémoire DRAM ou si l'objectif est de protéger les données présentes sur le disque dur. Les temps d'accès étant très différents (de l'ordre d'un facteur 100 en temps d'accès entre la mémoire DRAM et le disque dur), l'impact lié aux algorithmes de cryptographie ne sera donc pas le même. Si le temps d'accès au disque dur est de plusieurs dizaines de milliers de cycles d'horloge, alors ajouter quelques dizaines ou centaines de cycles pour une opération cryptographique supplémentaire n'est pas forcément critique du point de vue temporel. En revanche, si les données auxquelles on souhaite accéder ne sont pas dans le cache L1 (on parle de *cache miss*), il est nécessaire d'accéder aux données protégées dans le cache L2, ce qui provoque une pénalité de plusieurs centaines de

cycles d'horloge et peut induire une très forte dégradation des performances lors de l'exécution.

Les données présentes en mémoire peuvent être protégées de différentes façons ; il peut s'agir de leur confidentialité, de leur intégrité ou de leur authentification. Pour chaque propriété des algorithmes différents peuvent être utilisés, comme pour le cas de la protection du *boot*. Il est également possible d'utiliser des algorithmes permettant de vérifier plusieurs propriétés, par exemple un algorithme du type AES-GCM permet de garantir la confidentialité et l'authentification. Il est également important de définir une politique de sécurité concernant les données en mémoire. En effet, afin de réduire les pénalités liées à la protection des données (vitesse, surface de calcul, surface mémoire) il est essentiel de réfléchir à la façon dont les données sont protégées. Certaines données devant respecter des propriétés de confidentialité et d'intégrité, alors que d'autres doivent uniquement être protégées en intégrité. Ces différents points doivent être traités avec attention afin de réduire les surcoûts en mémoire et en temps liés à la sécurité. La protection des données nécessite donc une réflexion approfondie afin de définir les schémas cryptographiques à mettre en œuvre, ainsi que la politique de sécurité souhaitée.

Afin de s'assurer que les données n'ont pas été modifiées dans la mémoire externe, il faut garantir leur intégrité et donc calculer des empreintes permettant de vérifier leur état lors de leur utilisation. Pour garantir qu'il n'y a pas de rejeu d'anciennes données, il faut utiliser des informations temporelles associées aux données de telle sorte que lors de leur utilisation, il soit possible de vérifier leur « fraicheur » (c'est-à-dire vérifier qu'il s'agit bien des dernières données sauvegardées). Afin de garantir que les données n'ont pas été déplacées en mémoire, il est nécessaire de conserver une information permettant de faire le lien entre l'adresse de la donnée et la valeur de la donnée. Les mécanismes de protection mis en œuvre doivent donc intégrer ces différents paramètres au-delà des problématiques plus classiques de confidentialité, d'intégrité et d'authentification. Il est donc nécessaire de sauvegarder des informations supplémentaires de façon protégée contenant par exemple des informations temporelles ou des empreintes associées aux données.

Plusieurs solutions peuvent donc être mises en œuvre suivant le modèle de menaces. Par exemple, les scénarios suivants peuvent être considérés :

- L'objectif est de protéger les données en confidentialité. Dans ce cas il suffit d'utiliser une unité cryptographique avec un algorithme de chiffrement symétrique. Une clé secrète est sauvegardée dans l'unité cryptographique. Cette unité est par exemple placée entre le cache du processeur et la mémoire où sont sauvegardées les données. À chaque écriture en mémoire, les données sont chiffrées, et à chaque lecture de la mémoire, les données sont déchiffrées.
- L'objectif est de protéger les données en intégrité. Dans ce cas, on peut utiliser une unité cryptographique avec un algorithme de hachage. Une empreinte est associée à chaque bloc de données, et cette empreinte est sauvegardée à l'intérieur de l'unité cryptographique. L'empreinte est alors vérifiée lors de chaque lecture. Il est également possible d'utiliser un algorithme d'authentification de message (MAC en anglais, pour *Message Authentication Code*) : pour chaque bloc de données, un motif d'intégrité sera calculé et stocké en mémoire (ce qui introduit une expansion de la taille de la mémoire car il y a maintenant les données et leur code MAC à sauvegarder dans la mémoire). À la lecture, le motif d'intégrité est recalculé sur les données, et comparé au MAC stocké en mémoire. On utilise classiquement la construction HMAC, qui fait intervenir une fonction de hachage et une clé secrète, pour cet usage.
- L'objectif est de protéger les données en confidentialité et en intégrité. Dans ce cas il faut combiner les deux mécanismes précédents. L'unité cryptographique doit alors contenir un algorithme de chiffrement symétrique et par exemple une fonction de hachage avec clé.
- Si l'objectif est également de protéger la mémoire contre le rejeu et contre le déplacement des données en mémoire en plus de la confidentialité et de l'intégrité, il faut alors pour chaque

écriture en mémoire conserver une information temporelle (typiquement avec un compteur) et une information relative à l'adresse du bloc de données. Ces informations sont utilisées afin de calculer un code d'authentification associé au bloc de données à écrire en mémoire (elles vont donc être associées au code MAC). La mémoire à l'intérieur de l'unité cryptographique permet donc de mémoriser la valeur du compteur pour chaque segment d'adresses mémoire. Lors d'une lecture, comme précédemment, un nouveau code MAC est calculé pour le bloc de données chiffré en prenant en compte la valeur du compteur et l'adresse du bloc de données et ce dernier est comparé avec celui présent en mémoire.

La pénalité temporelle introduite par les algorithmes cryptographiques est un élément important car le temps d'accès est un paramètre essentiel, et toute opération cryptographique va introduire un surcoût temporel. Par exemple, l'utilisation combinée des algorithmes AES et SHA-2 peut nécessiter jusqu'à 74 cycles d'horloge afin de traiter les données alors qu'un algorithme du type AES-GCM peut réduire ce temps à 22 cycles d'horloge (voir le tableau 1 de l'article [6]). Il est également possible de paralléliser plus ou moins les algorithmes en fonction des performances souhaitées. Plus l'algorithme sera parallélisé, plus il sera rapide, mais en contrepartie il nécessitera davantage de surface (en taille de code ou en surface matérielle). Le tableau 3.2 de la thèse de Jérémie CRENNE [5] illustre ce point pour un algorithme de cryptographie.

Protection des communications à l'intérieur des circuits

Différentes architectures de communication existent dans les composants électroniques suivant le nombre d'éléments à connecter. Les architectures à base de bus et de réseaux sur puce (appelés *Network on Chip*, NoC, en anglais) correspondent aux solutions les plus classiques aujourd'hui [6, 7, 8, 9]. Lorsque le nombre d'éléments à connecter est supérieur à quelques dizaines d'unités une architecture à base de NoC s'impose notamment du fait des problématiques de congestion sur le bus.

Les architectures de communication à l'intérieur des circuits ne sont pas épargnées par les problématiques de sécurité et plusieurs attaques peuvent être mises en œuvre afin de compromettre les échanges de données. En effet, les menaces en confidentialité, en intégrité et en authentification existent également au niveau des communications. Un attaquant peut souhaiter changer le contenu d'une donnée (problème d'intégrité) ou bien lire en clair une donnée qui est secrète (problème de confidentialité), ou encore avoir accès à une donnée qui ne lui est pas destinée (problème d'authentification). Les données qui transistent sur un bus ou sur un NoC doivent donc être protégées en conséquence. Au-delà de ces menaces il existe d'autres points qui doivent être pris en compte, à savoir les accès en lecture ou en écriture (une unité sur un bus peut par exemple être autorisée à lire des données provenant d'une autre unité mais pas à y écrire des données). La taille des données est également un élement important de la sécurité afin de garantir que le volume de données transféré correspond bien à celui autorisé. Il peut également y avoir des problématiques de dénis de service, lorsqu'un attaquant souhaite saturer l'architecture de communication afin d'interdire tout échange de données.

Il faut donc prévoir des mécanismes de protection permettant de prendre en compte ces différentes menaces. Afin de protéger les communications, des unités de filtrage (intégrant éventuellement des mécanismes de cryptographie) sont généralement mises en œuvre. Ces unités sont placées en coupure entre les bus et les unités connectées aux bus [8]. La Figure 9 illustre l'utilisation des unités de filtrage pour une architecture composée de différentes ressources de calcul et de mémorisation. Pour les NoC les unités de filtrage peuvent être placées dans les interfaces réseau ou dans les routeurs. Ces unités permettent de garantir une politique de sécurité au niveau des accès aux ressources et aux

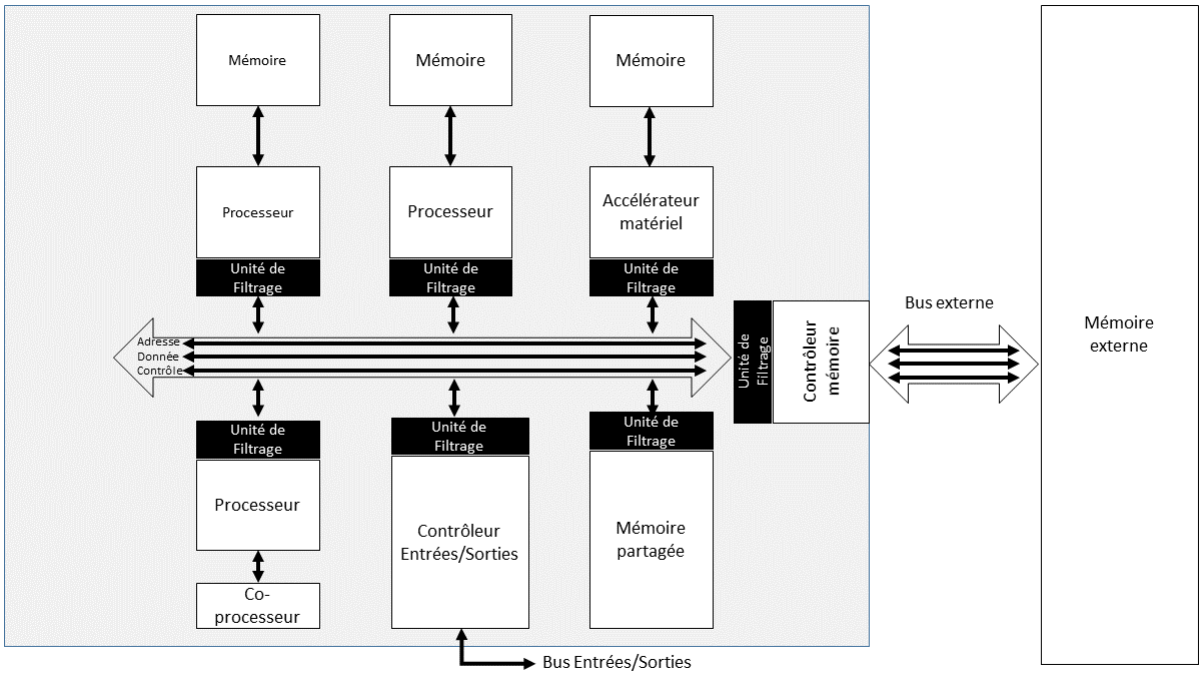

FIGURE 9 – Unités de filtrage en coupure [8].

données. La politique de sécurité permet, pour un système composé de différentes unités de calcul et de mémorisation, de préciser les unités qui sont autorisées à échanger des données (en lecture, en écriture, ou les deux), et celles pour lesquelles les échanges sont prohibés. Elle indique également les échanges pour lesquels des propriétés de confidentialité ou d'intégrité sont nécessaires. La définition de la politique de sécurité au sein d'une architecture de communication est donc un point très important, qui doit être traité avec beaucoup de rigueur (la section 3.1.4. de la thèse de Pascal COTRET illustre les politiques de sécurité [7]). En effet, la plupart des attaques débutent par une communication non autorisée.

Les politiques de sécurité mises en œuvre au sein des unités de filtrage sont généralement basées sur des tables contenant les règles de sécurité, de telle sorte que lors de chaque transfert, les règles associées aux données puissent être vérifiées (la section 3.2.2. de la thèse de Pascal COTRET illustre les règles de sécurité [7]). La protection des communications correspond donc à une activité de surveillance qui vise à contrôler certains paramètres pour déterminer si la requête qui est en train d'être analysée est conforme ou non au comportement normal. Par conséquent, l'unité de filtrage contient des mécanismes permettant de récupérer des informations relatives à la requête à analyser, mais également des informations concernant les différentes connexions autorisées ou non. Ces mécanismes sont utilisés pour filtrer et vérifier les accès entre les composants. Pour réaliser ces contrôles, les unités de filtrage utilisent deux types d'informations :

– Les informations relatives à la requête à analyser. Certains signaux de l'architecture de communication sont utilisés afin de connaître la nature de la requête. Ainsi, il est possible de savoir si la requête qui est en train d'être analysée est une lecture ou une écriture, et la taille des données à transmettre ou à recevoir.
– Une image des règles d'accès autorisées en conformité avec la politique de sécurité. Une erreur est détectée dès qu'il y a une différence entre les caractéristiques de la requête et les informations

extraites de la politique de sécurité.
Si des opérations cryptographiques sont également nécessaires, des ressources cryptographiques doivent être ajoutées au sein des unités de filtrage. Dans ce cas des solutions similaires à celles présentées pour la protection des mémoires peuvent être mises en œuvre. L'impact en termes de performance est également important à analyser. En effet, la vérification des règles d'accès aux données peut prendre quelques cycles d'horloge. De plus, de la même façon que pour la protection des mémoires, si des opérations de cryptographie sont nécessaires, ces temps dépendent des algorithmes de cryptographie utilisés (de quelques dizaines de cycles d'horloge à une centaine) [6, 7, 8].

La protection du *bitstream* pour les composants FPGA est également un enjeu important du point de vue de la sécurité [10, 11, 12, 13]. Lors de la mise sous tension d'un composant FPGA, ce dernier nécessite un fichier de configuration (appelé *bitstream*) lui permettant de charger son architecture. Ce fichier de configuration est potentiellement très sensible car il contient l'ensemble des informations présentes dans le composant FPGA au cours de son exécution. Il est donc très important de pouvoir garantir la protection du *bitstream*. En effet, plusieurs menaces sont envisageables tels que le clonage (un attaquant pourrait être intéressé par la configuration d'un composant FPGA afin de produire d'autres composants similaires), la retro-ingénierie (dans le but de connaître la configuration du composant) ou encore la contrefaçon (afin de produire à bas coût des composants similaires). Face à ces menaces, depuis une dizaine d'années, les fabricants de FPGA se sont emparés de cette question et proposent des solutions de plus en plus élaborées afin de garantir des propriétés de confidentialité, d'intégrité et d'authenticité du *bitstream*. Il peut également être nécessaire de garantir la version du *bitstream* pour interdire toute attaque du type rejeu (seule la dernière version du *bitstream* doit être chargeable). Afin de garantir ces propriétés les composants FPGA intègrent donc des ressources cryptographiques. Il contiennent également des générateurs de nombres aléatoires. Les solutions développées par les fabricants de FPGA intègrent également des contre-mesures aux attaques par canaux auxiliaires telles que les analyses en courant ou électromagnétique, présentées dans les fiches 1 à 3. Par exemple les composants FPGA IGLOO2 de Microsemi intègrent les éléments suivants [12] : une primitive de cryptographie asymétrique du type *Elliptical Curve Cryptography* (ECC), une primitive de cryptographie symétrique du type AES et une fonction de hachage du type SHA-2. Ces différents éléments sont intégrés au sein d'un bloc de sécurité.

Interfaces de mise au point

De nombreux composants proposent des interfaces de mise au point *(debug)*, permettant d'accéder de manière très privilégiée à leurs mémoires internes. Une des interfaces les plus répandues pour répondre à ce besoin est le connecteur JTAG (du nom du groupe, *Joint Test Action Group*, à l'origine du standard).

Si ce type d'interface est extrêmement utile dans la phase de réalisation et de mise au point d'un système, il est essentiel pour la sécurité de trouver un moyen de le désactiver avant la mise en production du système. En effet, un attaquant ayant accès à une interface de mise au point pourrait aisément lire les secrets présents dans les mémoires internes du composant, voire modifier directement son comportement.

La désactivation des interfaces de mise au point peut revêtir plusieurs formes : suppression complète de la fonctionnalité dans le composant, enfouissement des broches pour rendre l'accès plus difficile.

4.8 Matériels didactiques et références bibliographiques

[1] RAVI Srivaths, RAGHUNATHAN Anand, KOCHER Paul, HATTANGADY Sunil, *Security in embedded systems : Design challenges*, ACM Transactions on Embedded Computing Systems (TECS) Volume 3 Issue 3, August 2004 Pages 461-491 .

[2] Trusted Computing Group, http://www.trustedcomputinggroup.org/

[3] Infineon, http://www.infineon.com/cms/en/

[4] RUAN Xiaoyu, *Platform Embedded Security Technology Revealed*, August 18, 2014. Apress (open access) http://link.springer.com/book/10.1007/978-1-4302-6572-6

[5] CRENNE Jérémie, *Sécurité Haut-débit pour les Systèmes Embarqués à base de FPGAs* Thèse de l'Université de Bretagne-Sud, 9 décembre 2011, http://www.jeremiecrenne.com/

[6] COTRET Pascal, GOGNIAT Guy, *Protection des architectures hétérogènes sur FPGA : une approche par pare-feux matériels*, Techniques de l'Ingénieur, vol. IN175, pp. 1-10, 2014. https://pascalcotret.wordpress.com/

[7] COTRET Pascal, *Protection des architectures hétérogènes multiprocesseurs dans les systèmes embarqués. Une approche décentralisée basée sur des pare-feux matériels*, Thèse de l'Université de Bretagne-Sud, 11 décembre 2012. https://pascalcotret.wordpress.com/

[8] COTRET Pascal, GOGNIAT Guy, SEPULVEDA FLOREZ Martha Johanna, *Protection of heterogeneous architectures on FPGAs : An approach based on hardware firewalls*, Elsevier Microprocessors and Microsystems, en ligne 6 février 2016. http://arxiv.org/abs/1602.05106

[9] COBURN Joel, RAVI Srivaths, RAGHUNATHAN Anand, CHAKRADHAR Srimat, *SECA : Security-Enhanced Communication Architecture*, in Proc. 2005 Int. Conference on Compilers, Architectures and Synthesis for Embedded Systems (CASES), Sep. 2005, pp. 78-89.

[10] DEVIC Florian, *Securing embedded systems based on FPGA technologies*, Thèse de l'Université Montpellier II, 6 juillet 2012. http://www.univ-st-etienne.fr/salware/bibliography.html

[11] ALTERA, https://www.altera.com/

[12] XILINX, http://www.xilinx.com/

[13] Microsemi, http://www.microsemi.com/

5 Fiche 5 : Sécurité de la conception et protection de la propriété intellectuelle (IP)

5.1 Thématique

Thématique	Protection de la propriété intellectuelle et authentification des composants électroniques	**Numéro de fiche**	5	**Mise à jour**	03/03/2016

L'objectif de cette activité est de sensibiliser les étudiants à la problématique de protection des circuits et aux menaces possibles lors du cycle de conception d'un composant électronique. L'objet à protéger est généralement un circuit ou une partie d'un circuit (également appelée IP dans le domaine, pour *Intellectual Property*). Cette activité rappelle le cycle de conception et introduit les menaces à chacune des étapes du cycle. Elle présente également différentes solutions permettant de protéger les IP et les composants électroniques.

5.2 Thème des cours visés

– Conception de composants électroniques

5.3 Volume horaire

Cette activité nécessite une heure de présentation sous la forme d'un cours. Cette heure peut se décliner de la façon suivante :

– 15 minutes de rappel des enjeux liés à la contrefaçon ;
– 15 minutes de rappel du cycle de conception et des menaces accociées ;
– 30 minutes sur la protection des composants électroniques.

5.4 Prérequis / corequis

Cette activité s'intègre dans un cours de conception des composants électroniques. Elle peut être introduite dès que les étudiants ont des connaissances dans ce domaine. Elle permet aux étudiants de prendre conscience des problématiques de protections des IP et des composants face à la contrefaçon. Elle peut être présentée sous la forme d'une intervention du type cours magistral (CM).

Les solutions de protection peuvent faire appel à des algorithmes de cryptographie, aussi il peut être intéressant d'associer cette activité à une présentation de certains de ces algorithmes.

5.5 Objectifs pédagogiques

Cette activité vise à sensibiliser les étudiants aux problématiques de protection des IP et des composants électroniques face aux risques liés à la contrefaçon et au vol de la propriété intellectuelle.

5.6 Conseils pratiques

Cette activité est présentée sous la forme d'un cours. Dans un premier temps quelques éléments quantitatifs sont donnés afin de montrer les enjeux liés à la protection contre la contrefaçon et le vol de propriété intellectuelle. Le cycle de conception d'un composant électronique est ensuite présenté afin de faire ressortir ses différentes étapes et préciser les intervenants associés. Certaines étapes du cycle de conception sont généralement externalisées, aussi plusieurs entités interviennent tout au long de la conception, ce qui peut constituer des menaces. En effet, ces dernières sont possibles à chacune des étapes du cycle de conception, il est donc important d'en avoir conscience. Des mécanismes de protection permettant de garantir la sécurité des composants et des IP existent et sont introduits. Cette activité correspond à une sensibilisation, plusieurs références sont données afin d'approfondir des points qui pourraient susciter davantage d'intérêt de la part de l'enseignant et des étudiants.

5.7 Description

Préjudice lié à la contrefaçon

Le préjudice lié à la contrefaçon est une menace réelle et à ne pas négliger [1, 2, 6]. En effet, le marché de la contrefaçon est estimé à 7 % du marché mondial des semi-conducteurs. Pour les États-Unis le préjudice est évalué à 200 milliards de dollars par an, ce qui représente de l'ordre de 250 000 emplois perdus par an. Entre 2007 et 2010, la douane Américaine a saisi 5,6 millions de produits électroniques contrefaits. De nombreux cas de contrefaçon correspondent à des composants militaires et aéronautiques ce qui constituent des risques majeurs. Le cas d'une fausse usine NEC découverte en 2006, qui avait 50 références de produits du type home-cinéma, lecteur MP3, lecteur de DVD, clavier d'ordinateur, est un exemple marquant des situations qui existent aujourd'hui [3, 4]. Le cas de VisonTech aux États-Unis est également marquant. En effet, de 2006 à 2010, ce courtier américain a vendu plus de 60 000 contrefaçons de circuits intégrés [5]. Or, certains de ses clients appartenaient au domaine militaire (US Navy) et relevaient de systèmes radar ou de missiles *(Raytheon Missile System)*, ce qui pose des problèmes critiques en termes de protection militaire. Les conséquences sont nombreuses et peuvent être dramatiques pour les entreprises telles que des pertes de marchés avec comme conséquences sociales des pertes d'emplois, une insatisfaction des consommateurs et des dommages sur l'image de marque de l'entreprise, une non garantie de la sécurité des données et des systèmes sensibles ou encore une non garantie de la fiabilité opérationnelle des équipements. Il est donc indispensable de lutter contre la contrefaçon. La problématique de la protection de la conception et de la propriété intellectuelle est donc un enjeu majeur pour les entreprises. En effet, tous les types de circuits peuvent être touchés, depuis les transistors jusqu'aux mémoires, en passant par les composants configurables et les processeurs.

Cycle de conception et vulnérabilités

Afin de comprendre comment et où intervient la contrefaçon il est important d'avoir en tête le cycle de conception d'un composant électronique [7, 8]. En effet, différentes menaces pèsent sur les composants tout au long de ce cycle :
- le vol de propriété intellectuelle (IP) ;
- le vol de masques (*layout* en anglais), puces et circuits ;
- la copie / le clonage illégal ;

– la contrefaçon ;
– la rétro-ingénierie ;
– la modification de fonctionnalités (déblocage, DRM).

Le cycle de conception des composants électroniques est basé sur plusieurs étapes dont certaines doivent être externalisées dans des usines de production de circuits électroniques (appelées fonderies). Il existe des risques de vol de propriété intellectuelle au sein même de l'entreprise où travaille le concepteur, mais ce risque n'est pas l'objet principal de cette fiche qui s'intéresse davantage aux problèmes liés à la production du composant. Le concepteur aboutit à la fin de son étape de conception à une *netlist* du circuit, qui correspond à son architecture au niveau portes électroniques ou transistors. Cette *netlist* est ensuite envoyée chez le fondeur afin de produire le circuit, de le tester, de le mettre dans un boitier puis de le tester de nouveau avant de le retourner au client. Chaque étape comporte un risque de vol. En effet, si l'usine de production n'est pas de confiance, il peut y avoir le vol de la *netlist* afin d'en extraire la propriété intellectuelle (l'architecture complète du circuit ou certaines parties critiques ou secrètes), il peut y avoir également le vol du circuit avant ou après la mise en boitier afin d'aller vers de la surproduction. Le vol avant la mise en boitier permet, par exemple, aux attaquants de produire de faux circuits ayant les mêmes caractéristiques techniques que les circuits d'origine, mais sous une autre marque. Il existe également un risque en fin de vie du circuit qui, plutôt que d'être détruit, peut être reconditionné et revendu comme composant neuf. Ce cas pose de sérieux problèmes de fiabilité des composants, ce qui peut avoir des effets dramatiques. De telles situations ont été identifiées dans le domaine ferroviaire sur des fonctions liées à la sûreté de fonctionnement du système. Le cycle de conception ainsi que le cycle de vie d'un composant électronique subit donc différentes menaces que les concepteurs doivent pouvoir maitriser.

Protection des circuits

Des solutions permettent de protéger un circuit ou une partie d'un circuit (une IP) et cela au niveau des différentes étapes de conception, notamment avant l'étape d'externalisation qui vise à la production du circuit (avant l'étape de synthèse, pendant l'étape de synthèse et après l'étape de synthèse). Afin de protéger un circuit ou une IP le concepteur peut ajouter des marques (*watermarking* en anglais) [7, 8]. Ces marques doivent avoir les propriétés suivantes :
– pas d'impact sur les performances du circuit ;
– facile à vérifier ;
– difficile à détecter par un attaquant ;
– impossibilité de changer ou supprimer par l'attaquant ;
– surcoûts faibles (par exemple en surface ou en consommation) ;
– et bien sûr elles doivent permettre d'authentifier un circuit ou une IP de façon sûre.

Ces marques peuvent être ajoutées au niveau de la spécification des algorithmes en modifiant certains paramètres de l'algorithme sans que cela n'ait d'impact sur la fonctionnalité de l'application. Une autre approche possible consiste à modifier l'architecture du composant lors de l'étape de synthèse (par exemple lors de la synthèse comportementale ou de la synthèse logique). Dans ce cas l'architecture est légèrement modifiée afin de contenir une marque. Enfin, une marque peut être introduite après l'étape de synthèse en ajoutant par exemple des portes logiques permettant de produire une signature au niveau du composant. Comme indiqué précédemment, ces différentes marques ne doivent pas modifier les performances du circuit et sont introduites lors de la conception. Elles sont qualifiées de passives. D'autres approches basées sur des mécanismes dits actifs peuvent également être mises en œuvre. Dans ce cas la fonctionnalité du circuit peut être activée ou désactivée à l'aide de mécanismes spécifiques (par exemple en utilisant une clé secrète). L'acti-

vation est réalisée (à distance) en fin de process de fabrication. Ainsi, un circuit volé/copié avant activation n'est pas utilisable. Ces approches nécessitent un protocole cryptographique couplé à un blocage fonctionnel ; il peut s'agir par exemple du chiffrement de la logique / des machines à états (FSM en anglais, pour *Finite State Machine*), ou du chiffrement du chemin de données (bus, NoC). Ces différentes techniques permettent de garantir qu'un attaquant ne possédant pas l'information secrète (c'est-à-dire la clé) ne pourra pas utiliser le circuit ou l'IP.

Certaines solutions visent à faire de l'offuscation et bien que cela ne garantisse pas la sécurité du composant, plusieurs solutions industrielles utilisent ce mécanisme afin d'introduire un premier niveau de protection. L'offuscation (synonymes : assombrissement, opacification, obscurcissement) est une opération qui consiste à transformer une section de code ou un programme, de manière à le rendre totalement incompréhensible à un lecteur humain, même aidé d'outils informatiques. L'objectif est d'empêcher sa rétro-ingénierie. Ce type de technique va à l'encontre des règles de l'art dans le domaine de la programmation du logiciel et du matériel car il transforme un message (programme) clair (exécutable) en un autre message clair (toujours exécutable mais incompréhensible). Cette approche n'est pas une approche de sécurité mais vise à rendre plus complexe l'accès à la propriété intellectuelle. La qualité de l'offuscation est mesurée par la complexité et les performances de l'outil de désoffuscation. Des exemples de codes avant et après offuscation peuvent être trouvés dans [9] afin d'illustrer sa mise en œuvre.

5.8 Matériels didactiques et références bibliographiques

[1] PECHT Pecht,TIKU Sanjay, *Bogus ! Electronic manufacturing and consumers confront a rising tide of counterfeit electronics*, IEEE Spectrum, May 2006.

[2] GORMAN Celia , *Counterfeit Chips on the Rise*, IEEE Spectrum, June 2012.

[3] LAGUE David, *Next Step for Counterfeiters : Faking the Whole Compagny*, New York Times, May 2006. `http://www.nytimes.com/2006/05/01/technology/01pirate.html?pagewanted=all`

[4] CLARKE Peter, *Fake NEC compagny, says report*, EE Times, April 2006. `http://www.eetimes.com/electronics-news/4060352/Fake-NEC-company-found-says-report`

[5] *Chip counterfeiting case exposes defense supply chain flaw*, EE Times, October 2011. `http://www.eetimes.com/document.asp?doc_id=1266976`

[6] *Electronic Component Counterfeit Incidents Continue Record Pace as the US Department of Defense Set to Update Acquisition Rules*, IHS, October, 2012. `http://press.ihs.com/press-release/design-supply-chain/electronic-component-counterfeit-incidents-continue-record-pace-us`

[7] COLOMBIER Brice, BOSSUET Lilian, *Survey of hardware protection of design data for integrated circuits and intellectual properties*, IET Computers & Digital Techniques, 2014, Vol. 8, Iss. 6, pp. 274-287 doi : 10.1049/iet-cdt.2014.0028.

[8] Site web contenant de nombreuses références bibliographiques - SALWARE, *Salutary hardware design to fight against integrated circuit counterfeiting and theft*, 2013-2017. `http://www.univ-st-etienne.fr/salware/bibliography.html`

[9] MEYER-BAESE Uwe, *Obfuscation Tutorial*, `http://www.eng.fsu.edu/~umb/o4.htm`

Ce document pédagogique a été rédigé par un consortium regroupant des enseignants-chercheurs et des professionnels du secteur de la cybersécurité.

Version 1.7 — Février 2017