Contrefaçon et sabotage de microprocesseurs

Laurent Bloch[1]

Un composant peut-il être contrefait ?

La rétroconception (*reverse engineering*) est aussi vieille que la technique, et les polisseurs de pierre de toute l'Europe se sont probablement évertués à reconstituer les méthodes de fabrication des lames de silex du Grand-Pressigny, qui s'exportaient jusqu'en Hollande et aux Îles Anglo- Normandes. L'informatique n'a pas échappé à la règle. À l'époque des matériels à logique discrète c'était de l'ingénierie classique, tempérée par les brevets et les avocats chargés de les défendre ; on se rappellera les batailles homériques des années 1970 autour des interfaces d'IBM.

Depuis que les composants des ordinateurs, à commencer par le processeur, sont réalisés sous forme de circuits intégrés, on pouvait penser que la rétroconception par observation directe d'une puce de trois milliards de transistors sur deux centimètres carrés (soit un pas de dessin de 20 nanomètres) était impossible : n'aurait été possible que l'analyse des signaux (électromagnétiques en règle générale) émis par le composant à destination du monde extérieur, par ce que l'on nomme les canaux auxiliaires, observés en boîte noire, éventuellement manipulés par des techniques de *fuzzing*, qui consistent à perturber le composant en lui envoyant des données erronées (non conformes aux spécifications) et à scruter ses réponses.

Cette idée était erronée. Un article de quatre chercheurs de la Délégation générale à l'armement (DGA), Denis Réal, Julien Micolod, Jean-Claude Besset et Jean-Yves Guinamant, *La rétroconception de puces électroniques, le bras armé des attaques physiques*, publié dans le numéro hors-série n° 7 de la revue MISC, présente un panorama des avancées récentes dans le domaine de la rétroconception des puces électroniques. Disons d'emblée que ces techniques ne sont aujourd'hui pas employées contre des microprocesseurs d'usage général, tels que celui qui anime votre ordinateur, plutôt contre de plus petits systèmes, tels que des clés USB cryptographiques, mais demain… Et signalons aussi que ces techniques sont employées en conjonction avec les méthodes par canaux auxiliaires, pour extraire la clé privée du composant, ou des informations sur l'algorithme cryptographique utilisé. N'empêche, c'est possible.

C'est possible, mais pas à la portée du premier venu dans sa cuisine ou son garage.

Les échantillons doivent être préparés par des procédés physicochimiques adéquats de façon à dégager les couches intéressantes du composant.

1 Le parcours de Laurent Bloch, membre de l'ARCSI, auteur de cet article, est sur Linkedin : `https://www.linkedin.com/profile/view?id=514164`
Les Éditions Economica publieront en janvier 2015 son livre *La France et la Révolution cyberindustrielle*, et vous pouvez consulter son site ici : `http://www.laurentbloch.org/`

Pour observer des objets dont le motif élémentaire mesure une vingtaine de nanomètres, il convient de se munir d'un bon microscope électronique (un grossissement de 1 x 60 000 devrait suffire), ce qui limite déjà les ambitions. Prévoir aussi un banc de prise de vue avec un moteur pas à pas pour la capture d'images. Le budget matériel se monte déjà à quelques centaines de milliers d'euros.

Ainsi équipé, on peut obtenir une collection d'images utilisables par des logiciels d'assemblage et de vectorisation, ce qui procurera l'équivalent d'une photographie aérienne du composant, en quelque sorte.

Ce qui permet d'aller plus loin, c'est que la morphologie de ces composants est élaborée au moyen d'un langage de description, analogue à un langage de programmation, qui en génère le dessin sous une forme acceptable par les machines de fabrication. Les principaux langages de ce type sont VHDL et VERILOG. Ils engendrent des dessins très réguliers et répétitifs, ce qui facilite la reconnaissance des motifs, et donc la rétroconception. Il existe des logiciels de vérification des règles architecturales et électriques, qui permettent de traduire les images en liste d'interconnexions (netlist) acceptable par un compilateur VHDL ou VERILOG. L'analyse ainsi réalisée permet de comprendre le fonctionnement général du composant, compréhension qui sera affinée par l'observation des signaux électromagnétiques émis au cours de son fonctionnement, avec différents jeux de données.

Les analyses de ce type, si leur coût en réserve l'usage aux grandes entreprises, aux États développés, aux équipes de recherche correctement financées et aux cartels maffieux de grande envergure, ouvrent la voie à des attaques sur les couches basses du matériel, très dangereuses parce qu'elles contournent toutes les protections par matériel ou par logiciel.

En s'équipant d'un microscope électronique à balayage (grossissement de 1 x 60 000), d'un banc de prise de vue, de produits chimiques pour accéder aux couches successives du circuit, de logiciels d'assemblage et de vectorisation pour établir sa cartographie, et, *in fine*, de logiciels de vérification des règles architecturales et électriques, qui permettent de traduire les images en liste d'interconnexions (*netlist*) acceptable par un compilateur VHDL ou VERILOG (les langages de description de micro-architectures), il est donc possible de reconstituer l'architecture et le mode de fonctionnement d'un microprocesseur pas trop complexe, et ainsi de le contrefaire.

Un composant peut-il être saboté ?

Georg T. Becker, Francesco Regazzoni, Christof Paar et Wayne P. Burleson de l'université du Massachusetts à Amherst ont conçu une méthode pour effectuer, en quelque sorte, l'opération inverse, modifier subrepticement le fonctionnement d'un microprocesseur aussi complexe que l'on veut.

Le type d'attaque proposé est particulièrement dangereux, parce que pratiquement indétectable, et si le matériel est compromis, on ne peut plus se fier à aucun logiciel. Les cibles les plus prometteuses, selon les auteurs, seraient les générateurs de nombres pseudo-aléatoires utilisés par les opérations de calcul cryptographique (désormais incorporées à tous les processeurs modernes) pour réduire l'entropie du tirage et affaiblir le chiffrement ; ainsi, l'utilisateur croirait tirer un nombre au hasard parmi les nombres de

128 chiffres binaires, mais en réalité il tirerait dans un ensemble beaucoup plus petit, par exemple celui des nombres de 56 chiffres binaires ; ce dernier ensemble est suffisamment petit pour qu'un assaillant puisse se livrer à une attaque par force brute. Cette méthode de sabotage permet en outre, comme nous allons le voir, de s'attaquer à un des maillons les plus faibles de la chaîne de fabrication du processeur, ce qui est un autre avantage pour les pirates.

Plaçons-nous à l'étape du processus où le producteur de microprocesseurs dispose d'une maquette du processeur qu'il veut proposer à ses clients, sous la forme d'un programme exécutable, issu de la compilation d'un texte écrit dans un langage de description (HDL), qui va permettre de piloter les machines de fabrication, steppers ou scanners et autres équipements de photolithographie. C'est entre l'envoi de la maquette du circuit à la fonderie et la sortie des pièces de la chaîne de fabrication que se présentent les meilleures occasions pour l'introduction d'une modification malveillante.

Une première piste à laquelle pourrait songer le pirate serait d'ajouter au composant un circuit supplémentaire de son cru, qui détournerait le fonctionnement nominal du composant selon ses plans, ou de modifier une partie du composant. Cela suppose bien sûr des failles de sécurité au cours de la fabrication, dans l'usine. Cette piste semble peu prometteuse, parce que les modifications effectuées seraient détectables, soit par un examen au microscope, soit parce que les composants modernes sont généralement pourvus de circuits d'auto-vérification (*Built-In Self-Test*, BIST) capables de détecter un comportement non conforme aux spécifications nominales.

Jouer sur le dopage des composants

La méthode proposée et expérimentée par les auteurs de notre article consiste à modifier le comportement d'un composant en modifiant le dopage (c'est-à-dire le taux d'impuretés) du matériau en certains points précis pour établir ou au contraire supprimer des contacts électriques, et ainsi modifier le comportement du circuit.

Cette méthode innovante présente des avantages décisifs par rapport à celles évoquées ci-dessus : modification frauduleuse de la description du composant, ou insertion frauduleuse d'un circuit parasite, au niveau HDL ; ces deux types de sabotages sont en effet détectables, tant par examen au microscope électronique que par exécution des procédures de test algorithmique.

L'idée centrale de la méthode décrite par nos auteurs est la suivante : une porte logique du schéma original est modifiée en certains points précis par introduction de porteurs de charges différents de ceux du schéma. Ces modifications altèrent le comportement de la porte d'une façon prévisible. Une telle modification est indétectable au microscope électronique, et les deux exemples de réalisation par les auteurs de l'article ont résisté aux procédures de vérification prévues par les industriels.

Exemple 1 : sabotage du générateur de nombres pseudo-aléatoires de l'architecture Ivy Bridge d'Intel

La conception d'un bon générateur de nombres pseudo-aléatoires a toujours été un art difficile et plein de pièges, et l'histoire de l'informatique est pleine d'échecs dans ce domaine. Cette question est particulièrement sensible aujourd'hui, parce que beaucoup

de méthodes de chiffrement et de signature électronique reposent sur le générateur de nombres pseudo-aléatoires du système sous-jacent.

Un bon générateur de nombres pseudo-aléatoires, invoqué un grand nombre de fois, doit produire des nombres uniformément répartis sur un intervalle aussi large que possible. Le générateur de nombres pseudo-aléatoires de l'architecture Ivy Bridge d'Intel, dont il est question ici, produit des nombres de 128 chiffres binaires, soit compris entre 0 et 2^{128}-1, en d'autres termes entre 0 et 340 282 366 920 938 463 463 374 607 431 768 211 455 ; la taille de cet intervalle détermine ce que l'on nomme l'entropie du générateur. Moins le nombre est prédictible (plus il s'approche de l'aléatoire idéal), plus l'entropie est élevée (plus la quantité d'information qu'il fournit est grande).

Aussi étrange que cela puisse paraître, un bon générateur de nombres pseudo-aléatoires doit aussi être déterministe, c'est-à-dire que si on lui fournit deux fois de suite la même valeur initiale, il doit produire deux fois de suite la même suite de résultats. C'est pourquoi le fonctionnement d'un tel générateur est généralement initialisé à partir d'une source d'entropie externe, par exemple une valeur calculée à partir des déplacements erratiques de la souris, ou des turbulences engendrées par le système de refroidissement du processeur.

Deviner par force brute un nombre pseudo-aléatoire tiré dans cet intervalle consiste à essayer un par un tous les nombres consécutifs, jusqu'à tomber sur celui que l'on cherche… par hasard. Avec l'intervalle que nous venons de décrire et les processeurs actuels, le temps qui reste à exister au système solaire n'y suffirait pas. Réduire l'entropie d'un générateur est donc intéressant pour un attaquant qui cherche à casser un système de chiffrement.

Le générateur de nombres pseudo-aléatoires de l'architecture Ivy Bridge d'Intel est considéré comme très sûr, il est très bien documenté et doté de procédures de test et de vérification, internes et externes, conformes au meilleur état de l'art. Les procédures de vérification interne (BIST) sont effectuées à chaque démarrage du système.

L'article décrit le procédé par lequel, en modifiant le dopage d'une porte, à une échelle très petite, une variable du calcul est contrainte à une valeur constante, cependant que parmi les 128 chiffres d'une autre variable, *n* sont maintenus constants, *n* étant choisi au gré de l'attaquant. Le nombre pseudo-aléatoire obtenu possède toutes les qualités requises par la suite de tests imposée par le *National Institute of Standards* (NIST SP800-90), simplement il est choisi dans un intervalle arbitrairement petit, tel que l'attaquant puisse le trouver par force brute. Les auteurs ont veillé à ce que le calcul consomme la même quantité d'énergie électrique que celle utilisée par le circuit non falsifié, ce qui fait obstacle à la surveillance sur les canaux auxiliaires (cf. ci-dessous). Cette falsification est donc très difficile à détecter, les résultats du calcul effectué par le circuit falsifié présentent tous les caractères statistiques désirés et résistent à toutes les procédures de vérification, et l'attaque permet l'accès à la clé secrète de la cible.

Exemple 2. Attaque par canaux auxiliaires : créer (furtivement) le canal

Un algorithme, cryptographique ou autre, se comporte différemment selon la nature des données qui lui sont soumises : le nombre d'instructions exécutées diffère selon que la clé de chiffrement est bonne ou mauvaise, il en résulte que les temps d'exécution ne

sont pas les mêmes, non plus que la consommation électrique, ni la chaleur dissipée. Ces phénomènes physiques sont nommés canaux auxiliaires (side-channels en anglais), et leur observation sur un grand nombre de cas peut, dans certains cas, donner accès à un secret, tel qu'une clé de déchiffrement. C'est une attaque par canal auxiliaire.

Pour se prémunir d'une attaque par canal auxiliaire, les auteurs de logiciels de chiffrement et de déchiffrement adaptent leurs algorithmes de sorte qu'ils exhibent le même comportement extérieur quel que soit le calcul effectué, par exemple, s'il s'agit de traiter le signal a, ils traitent toujours a et non a, ainsi le canal auxiliaire montre toujours la même chose, il est donc inutilisable.

Il est donc intéressant pour un attaquant d'ouvrir un canal auxiliaire là où il n'y en a pas. C'est ce qu'ont fait nos auteurs, en s'attaquant à un circuit de chiffrement AES conçu pour résister à de telles attaques, par les subterfuges suivants : toujours effectuer simultanément le calcul pour un signal a et pour son complémentaire non a, réinitialisation des portes logiques à chaque cycle d'horloge, randomisation de certains calculs. Ces méthodes de dissimulation reposent sur une combinaison assez complexe de portes logiques, pour la description de laquelle le lecteur peut se reporter à l'article original *Stealthy Dopant-Level Hardware Trojans?*. Nos auteurs, dans le rôle de l'attaquant, ont « simplifié » le circuit en inhibant certaines zones des transistors concernés, toujours en modifiant le dopage de zones sensibles. Ils ont ainsi créé les conditions pour une consommation électrique dépendante des données, et prédictible. Ils ont simulé le circuit saboté, et comparé son comportement à celui du circuit normal. L'analyse du canal auxiliaire montre effectivement qu'il permet de récupérer la clé secrète calculée par le circuit, et qu'en outre le circuit est résistant aux attaques par canal auxiliaire triviales (c'est-à-dire celles des autres).

Ce travail spectaculaire et remarquable place très haut la barre à franchir pour se prémunir d'attaques contre le matériel, attaques qui réduisent à néant toute sécurité du logiciel. Heureusement, Messieurs Georg T. Becker, Francesco Regazzoni, Christof Paar et Wayne P. Burleson se proposent, au cours de leurs travaux à venir, de réfléchir à des moyens de détection de telles attaques. Espérons qu'ils trouveront rapidement !

Un composant peut-il être certifié ?

Pourquoi pas ? De ce qui précède on peut comprendre que la certification d'un processeur suppose la certification des logiciels utilisés, celles des entreprises qui ont fourni les schémas de certains sous-ensembles, celle de l'entreprise qui réalise l'assemblage final, enfin celle de la fonderie. Bien sûr c'est plus simple si l'on considère une entreprise qui contrôle le processus de bout en bout : il n'y a plus guère qu'Intel qui soit dans ce cas pour des processeurs d'usage général, mais il y a sans doute plus de choix pour des processeurs spécialisés, par exemple à usage militaire, et c'est un sujet pour lequel une intervention étatique ne serait peut-être pas inopportune.