Optimisation de contre-mesure à l'insertion de Hardware Trojan

Jonathan Fontaine[1], Lilia Zaourar[1], Roselyne Chotin[2]

[1] CEA, LIST ; Computing and Design Environment Laboratory 91191 Gif sur Yvette, France
{jonathan.fontaine, lilia.zaourar}@cea.fr, roselyne.chotin@lip6.fr
[2] Sorbonne Université, CNRS, LIP6, F75005, Paris, France

Mots-clés : *Logic locking, Sécurité des circuits intégrés, Clique*

1 Introduction

De nos jours, la complexité croissante des appareils électroniques a engendré une augmentation de leur coût de production. Par conséquent, divers agents externes assurent une partie de la chaîne de production d'un *Circuit Intégré* (*CI*). Ces derniers, ne dépendant pas de la compagnie à l'origine de la conception du *CI*, peuvent insérer des composants électroniques indésirables à des fins malveillantes, appelés des *Hardwares Trojan*[1] (*HT*). Ceci pose un problème de sécurité majeur, surtout pour les *CI* utilisés dans les domaines d'importance vitale tels que le transport, la santé ou le militaire [3]. Ces problèmes vont de la fuite d'information à la détérioration accélérée du matériel, en passant par le déni de service.

Le magazine Bloomberg Businessweek titre « The Big Hack : How China Used a Tiny Chip. to Infiltrate U.S. Companies », qui nous informe que des petits modules électroniques inconnus avaient été ajoutés dans des composants pour serveur de très grandes entreprises américaines.

Pour se prémunir de ces composants malicieux, il existe plusieurs contre-mesures, classifiées en deux catégories : prévention et détection. La détection est difficile car les *HT* sont des petits modules furtifs, et peuvent prendre différentes formes. La prévention, quant à elle, modifie la conception du *CI*. Malgré le surcoût engendré, ces méthodes restent efficaces. Elles sont appelées *Design for Hardware Trust* (DfHT). Elles ont chacune leur coût et leurs effets sur la sécurité. Dans ce travail nous utilisons la méthode de *logic locking*.

Le principe est de verrouiller le *CI* avec une clé numérique, seulement connue du concepteur (et d'un tiers de confiance). Ce qui est efficace contre les *HT* comme présenté dans [2]. L'utilisation du *logic locking* dans la lutte contre l'insertion des *HT* est issue de [2]. Le principe étant que chaque bit de la clé numérique soit relié à une nouvelle porte logique du circuit, appelé porte clé, qui occulte la fonction logique du circuit. Ceci rend plus complexe l'insertion du composant malicieux. Les objectifs pour rendre le *logic locking* efficace sont la sécurité, la non dégradation des performances du circuit (chemin critique, consommation, surface).

L'objectif de ce travail est d'optimiser l'insertion du *logic locking* afin de maximiser la sécurité du *CI*. Nous présentons ici la formulation mathématique ainsi que quelques pistes de résolution.

2 Modélisation du problème

Soit un *CI* représenté sous forme de *netlist*, qui est une interconnexion de portes logiques reliant les entrées et sorties du circuit. Nous modélisons cette *netlist* sous forme de graphe, $G = (V, E)$, avec V l'ensemble des portes logiques et des entrées du circuit et E les signaux entre les portes logiques. Nous supposons ici d'une part qu'un chemin de signal non occulté entre une entrée et un sommet, permet de museler une information inconnue sur ce sommet. D'autre part que tous les sommets propagent de la même façon une information inconnue.

La mesure de sécurité que nous allons utiliser est le *pairwise secure* [4]. Elle consiste à déterminer une notion de muselage entre les couples de portes clés. Une porte clé k_1 est muselée

1. appelé chevaux de Troie matériel en français

pour déterminer k_2, s'il existe un pattern d'entrée qui permet d'observer le bit clé de k_2 sans interférence du bit clé de k_1. Un couple est *pairwise secure* lorsque il n'y a ni muselage d'une porte clé k_1 vers une autre k_2, ni de k_2 vers k_1.

Nous imposons une augmentation maximale de surface par une limitation du nombre de portes clés au travers d'une valeur K. Notons $I(n)$ l'ensemble des indices d'une matrice carrée de taille n, et $T(n)$ l'ensemble des indices d'une matrice triangulaire supérieure de taille n. $precs(n)$ et $sortie(G)$ désignent les prédécesseurs du sommet n dans G et les sorties de G.

Notre objectif est de maximiser la sécurité, dans notre modèle, cela revient à maximiser la plus grande clique (1). Soit la variable C_i indiquant si la i-ème porte clé fait partie de la plus grande clique. La variable A_n^i permet de lier la i-ème porte clé au n-ième sommet du graphe. La variable $NM^{(k,k')}$ représente le non muselage de la porte clé k' par rapport à la porte clé k. La variable $PW^{(k,k')}$ indique si le couple (k, k') est *pairwise secure*. La variable $X_n^{(k,k')}$ indique si dans la recherche de muselage de la porte clé k', au sommet n, la porte clé k a une influence logique sur ce sommet. La variable $Y_n^{(k,k')}$ indique si la porte clé k' à une influence logique sur le sommet n, dans la recherche de muselage de la porte clé k'. L'ensemble des variables du problème sont binaires.

$$
P_{LL} = \begin{cases}
\max \displaystyle\sum_{i\in\{K_1,\dots,K_K\}} C_i & & (1)\\
\underline{s.t.}\ \ C_k + C_{k'} \le 1 + PW^{(k,k')} & \forall (k,k') \in T(K) & (2)\\
\displaystyle\sum_{n\in V} A_n^i \le 1 & \forall i \in \{K_1,\dots,K_K\} & (3)\\
\displaystyle\sum_{i\in\{K_1,\dots,K_K\}} A_n^i \le 1 & \forall n \in V & (4)\\
X_n^{(k,k')} = A_n^k \vee \left(\displaystyle\bigvee_{p\in precs(n)} X_p^{(k,k')} \bigwedge_{p\in precs(n)} \neg Y_p^{(k,k')} \right) & \forall (k,k') \in I(K),\ \forall n \in V & (5)\\
Y_n^{(k,k')} = A_n^{k'} \vee \left(\displaystyle\bigwedge_{p\in precs(n)} (X_p^{(k,k')} \vee Y_p^{(k,k')}) \wedge \bigvee_{p\in precs(n)} Y_p^{(k,k')} \right) & \forall (k,k') \in I(K),\ \forall n \in V & (6)\\
NM^{(k,k')} = \left(\displaystyle\bigvee_{n\in V} A_n^k \wedge \bigvee_{n\in V} A_n^{k'} \right) \wedge \bigwedge_{s\in sortie(G)} X_s^{(k,k')} \implies Y_s^{(k,k')} & \forall (k,k') \in I(K) & (7)\\
PW^{(k,k')} = NM^{(k,k')} \wedge NM^{(k',k)} & \forall (k,k') \in T(K) & (8)
\end{cases}
$$

(2) impose que deux éléments d'une clique peuvent être simultanément sélectionnés s'ils sont *pairwise secure*. (3) et (4) imposent qu'un seul sommet peut être associé à une porte clé et inversement. (5) impose la propagation de l'inconnue à propager. (6) exprime la propagation de l'inconnue à inhiber. (7) modélise la notion de non muselage pour un couple de portes clés. (8) modélise la notion de *pairwise secure*.

3 Résolution

Pour résoudre ce modèle, nous avons en premier lieu linéarisé les contraintes (5) - (8) afin d'utiliser un solveur classique de programmation linéaire en nombres entiers. Cependant le modèle est trop gros pour traiter des circuits de 1000 portes logiques. Nous avons ensuite développé une heuristique. Pour chaque couple, on calcule la notion de *pairwise secure*, grâce aux formules précédentes, dans une matrice triangulaire supérieure. Ensuite on récupère toutes les cliques maximales via l'algorithme de Bron-Kerbosch [1]. Enfin, on sélectionne les plus grandes cliques, jusqu'à ce que notre limite de portes clés soit atteinte. Nous vous présenterons les résultats numériques obtenus, ainsi que des perspectives d'amélioration.

Références

[1] C. Bron et J. Kerbosch. "Algorithm 457 : Finding All Cliques of an Undirected Graph". In : *Commun. ACM* (1973).

[2] S. Dupuis et al. "A novel hardware logic encryption technique for thwarting illegal overproduction and Hardware Trojans". In : *IEEE IOLTS*. 2014.

[3] K. Xiao et al. "Hardware Trojans : Lessons Learned after One Decade of Research". In : *ACM Transactions on Design Automation of Electronic Systems* (2016).

[4] M. Yasin et al. "On Improving the Security of Logic Locking". In : *IEEE TCADICS* (2016).