

L'auto-test d'un cœur de chiffrement AES

Marion DOULCIER, Marie-Lise FLOTTES, Bruno ROUZEYRE

Université Montpellier II / CNRS

Laboratoire LIRMM – UMR 5506 CNRS

161 rue Ada

34932 Montpellier Cedex 5

Email : marion.doucier@lirmm.fr

-Résumé

L'auto-test intégré est une solution de test matériel particulièrement bien adaptée au cas des systèmes intégrés sécuritaires puisqu'il limite l'accès aux données confidentielles traitées par le dispositif. Ce papier présente une solution architecturale innovante et performante d'auto-test pour un cœur de chiffrement AES (Advanced Encryption Standard).

1. Introduction

Le principe de la cryptographie est de s'assurer que les données sensibles stockées ou échangées via des réseaux soient uniquement lisibles par le destinataire et qu'en aucun cas elles ne puissent être lues par une autre personne. Des mécanismes cryptographiques sont implantés dans les circuits afin de préserver la confidentialité des données. Ces circuits font partie de notre quotidien, leurs utilisations sont diverses : téléphonie, télévision, sécurisation des accès, cartes de paiement, carte vitale, passeport électronique...

D'un point de vue de la fabrication de ces circuits, l'évolution technologique se traduit par une diminution de la taille des transistors, une densité d'intégration plus importante et une fréquence de fonctionnement des circuits qui ne cesse de croître. Ces évolutions rendent le test de production des systèmes intégrés de plus en plus difficile. L'activité de test concerne tous les composants microélectroniques produits mais requière une attention toute particulière dans le contexte des circuits sécurisés. En effet, tout défaut non décelé peut engendrer une faille sécuritaire. D'autre part, les techniques de test habituelles consistent à augmenter considérablement la contrôlabilité et l'observabilité des états internes du circuit et sont donc contradictoires avec les contraintes liées à la sécurité numérique qui demandent à ce que les données manipulées (information confidentielles, clés de codage) ne puissent pas être facilement contrôlées ou observées par un attaquant.

Dans la section 2, nous présenterons les différentes techniques de test, leurs avantages et leurs inconvénients. L'algorithme de chiffrement AES, son implantation et son fonctionnement en mode d'auto-test sont présentés en section 3. Dans la section 4, nous présenterons l'étude menée pour déterminer la longueur minimale de la séquence d'auto-test pour l'AES. La section 5 présente

nos résultats expérimentaux. Et la section 6 conclue le papier.

2. Techniques de Test des circuits sécurisés

La technique de test externe basée sur le « scan-path » par exemple [1], est connue pour fournir une très bonne qualité de test mais permet aussi de monter une attaque spécifique basée sur l'utilisation frauduleuse du scan path. Des attaques ont été démontrées dans [2] et [3] respectivement dans le cas d'un cœur de chiffrement DES (Data Encryption Standard) et AES. Des contremesures ont été proposées qui consistent à modifier l'implantation classique d'un scan path, induisant surface additionnelle et modification du flot de conception [4].

Les techniques de test intégré BIST (« Built-In Self Test ») ne fournissent pas d'accès depuis l'extérieur aux données traitées en interne, elles sont donc plus appropriées dans le cas de circuits sécurisés. Deux approches de BIST sont possibles : une dite de test en ligne et l'autre dite de test hors ligne.

Le test en ligne consiste à tester le circuit durant son fonctionnement normal (pas de mode spécifique de test). Plusieurs solutions ont été proposées dans ce contexte [5] [6] [7]. Le principe consiste à vérifier la parité des données entrantes et/ou sortantes du circuit en la comparant avec une parité attendue. Ce type d'approche est couramment utilisé pour s'assurer de la fiabilité d'un dispositif en vérifiant la validité des données traitées en cours de fonctionnement. Il ne permet pas d'assurer un test de production puisqu'il ne vise pas à déceler la présence d'une faute avant de délivrer le circuit à l'utilisateur.

Pour ce qui concerne le BIST hors ligne, cette approche est utilisable comme test en fin de production pour trier les circuits sains des circuits fautifs, et comme solution de test durant la vie du circuit. Elle ne peut toutefois être appliquée concurremment au fonctionnement normal puisqu'elle nécessite l'application de données de test spécifiques et non de données quelconques de fonctionnement. Le BIST hors ligne implique donc la création d'un nouveau mode de fonctionnement dit mode test. Ce mode consiste à appliquer une séquence de vecteurs de test au circuit et à comparer les réponses du circuit avec celles attendues.

Cette approche induit donc généralement l'insertion de ressources de test supplémentaires ainsi qu'une adaptation du contrôleur pour commuter du mode normal

au mode test et vice versa. L'augmentation de surface liée à cette approche à l'effet néfaste d'augmenter les probabilités d'apparition d'un défaut puisque la présence de défauts augmente avec la surface de silicium utilisée. Elle n'est viable que si la surface supplémentaire requise est très faible par rapport au circuit sous test.

La technique de BIST la plus couramment employée consiste à appliquer une séquence de vecteurs de test pseudo-aléatoires générés à l'aide d'un LFSR (Linear Feedback Shift Register) ou d'un automate cellulaire puis à valider la signature obtenue à l'aide d'un MISR (Multiple Input Signature Register) en sortie après compression des réponses à chacun des vecteurs. La génération aléatoire de données et la compression de signature présentent l'avantage d'être implantables avec des ressources de très faible complexité mais induisent néanmoins une surface additionnelle non négligeable.

Nous présentons ici une méthode d'auto-test intégré pour cœur de chiffrement AES basée sur un rebouclage de la structure. La qualité de test obtenue est exprimée en terme de taux de couverture des fautes simples de collage (nombre de fautes détectables par le mode test / nombre total de fautes pouvant affecter le circuit). Cette étude s'inscrit dans le cadre d'une thèse portant sur le test, la testabilité et la fiabilité de systèmes sécurisés numériques et vient en complément de résultats déjà obtenus en ce qui concerne l'utilisation d'un cœur AES en tant que générateur de vecteurs de test pseudo-aléatoires et analyseur de signature pour d'autres cœurs du même système [8].

3. L'algorithme de chiffrement AES

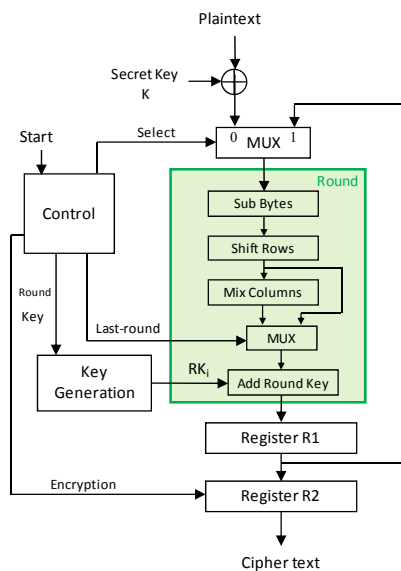


Figure 1 : L'algorithme AES

L'algorithme de chiffrement AES a été développé par deux chercheurs belges - Vincent Rijmen et Joan Daemen. Cet algorithme a été officiellement approuvé comme standard le 6 décembre 2001 [9]. L'AES est un algorithme de chiffrement par bloc, il permet de chiffrer un texte en clair (Plaintext) constitué de 128 bits de données à l'aide d'une clef secrète constituée de 128, 192 ou 256 bits. L'algorithme se compose principalement d'un module "ronde" qui sera itéré 10, 12 ou 14 fois suivant la taille de la clef secrète utilisée. Une ronde d'AES est constituée de 4 opérations : Sub Bytes, Shift

Rows, Mix Columns et Add Round Key. L'opération Mix Columns n'est pas effectuée lors de la dernière ronde. Nous considérerons par la suite que la taille de la clef est 128 bits. Le principe du chiffrement est présenté sur la figure 1, le module ronde est représenté par la partie colorée.

Ces opérations traitent les 128 bits de données sous forme de matrice 4x4 dont les éléments sont des octets.

L'opération Sub Bytes est une substitution non-linéaire d'octet à l'aide d'une table appelée S-box. La modification engendrée par la S-box, traitant 8 bits de données, est généralement pré-calculée pour l'ensemble des valeurs entrées (2^8) et donc la S-box peut être implanté sous forme de RAM ou de logique combinatoire.

L'opération Shift Rows est un décalage circulaire des lignes de données de la matrice. La première ligne ne subit pas de décalage, la seconde, la troisième et la quatrième se décalent de façon circulaire vers la gauche respectivement de un, deux et trois octet.

Mix Columns est une opération de multiplication entre une matrice d'éléments connus et la matrice de données résultant de l'opération Shift Rows.

L'opération Add Round Key est une opération de ou-exclusif entre la matrice de données et la clef de ronde. Les dix clefs de ronde (une pour chaque rondes) sont calculées à partir de la clef secrète.

Dans [10], il est montré que le test pseudo-aléatoire de cœur de chiffrement est une technique efficace. Des taux de couverture élevés sont obtenus avec un faible nombre de vecteurs de test pseudo-aléatoires. Une des raisons est que les opérations de chiffrement traditionnelles xor, substitution, modulo... sont facilement testables avec des données aléatoires. De plus, ces opérations ont une propriété intéressante qui est de laisser se propager les données aléatoires à travers le circuit.

La procédure de test consiste à appliquer un texte en clair et une clef secrète à l'AES est de laisser celui-ci effectué un certain nombre de rondes. Etant donné que l'AES génère des vecteurs aléatoires à la fin de chaque ronde [8], l'idée est de les réinjecter dans le circuit pour tester le circuit lui-même. Le nombre de rondes à réaliser est équivalent au nombre de vecteur de test, la valeur minimale du nombre de rondes à effectuer pour tester toutes les fautes de collage est présentée dans la section 4.

Les modifications à apporter à la structure pour implanter cette procédure d'auto-test se résument à agir sur le module de contrôle et sur le module de génération des clefs de ronde. En fonctionnement normal, un texte en clair est appliqué puis 10 rondes sont effectuées. La modification consiste à pouvoir réaliser plus de 10 rondes, mais toujours en suivant le même principe à savoir que lors des rondes multiples de 10, l'opération Mix Columns n'est pas effectuée. En résumé, le signal *Select* (voir figure 1) doit être à mis 0 au début puis à 1 durant toute la phase d'auto-test, ceci à fin de pouvoir appliquer au module de ronde le nombre de vecteurs suffisant pour la tester, et le signal *Encryption* doit être actif à la fin de la phase d'auto-test afin d'obtenir la réponse du circuit. Le module de génération des clefs de rondes est quant à lui modifié, de sorte qu'après avoir calculé les 10 clefs de ronde à partir de la clef secrète, la

dixième clef de ronde devient la clef à partir de laquelle 10 nouvelles clefs de rondes vont être calculées et ainsi de suite. Ainsi pour chacune des rondes à réaliser pendant la phase d'auto-test une clef de ronde différente sera utilisée.

4. Longueur minimale de la séquence de test

Cette section présente les résultats théoriques sur l'étude de la longueur minimale de la séquence de vecteurs de test pour garantir l'auto-test de l'AES.

Etant donné que 83% de la surface totale de l'AES est constituée de l'implantation de l'opération Sub Bytes (16 S-boxes 8-bits), nous nous sommes en premier intéressés à la testabilité d'une S-box. Les S-boxes sont décrites en VHDL et synthétisées sous forme de logique combinatoire à l'aide de la librairie CMOS AMS 0,35µm.

Pour une S-box, un dictionnaire de fautes, association faute et vecteur(s) détectant(s), a été établi et le nombre minimal de vecteurs de test requis pour tester toute la S-box a été calculé. Ces résultats ont été obtenus à l'aide d'un outil développé au laboratoire, celui-ci est une association entre un simulateur de fautes et une heuristique de recherche de « recouvrement minimal ».

La longueur minimale de la séquence de vecteurs de test déterministes, pour obtenir un taux de couverture de 100%, est de 203 vecteurs ceci pour une S-box à 8 bits d'entrée. En résumé parmi les 2^8 (256) vecteurs possibles en entrée de la Sbox, 203 d'entre eux sont nécessaires pour tester complètement la Sbox. Il se peut qu'il existe plusieurs ensembles de 203 vecteurs parmi les 256 qui permettent de tester la S-box. Nous considérons par la suite une seule de ces combinaisons, (approche pessimiste).

Nous avons ensuite déterminé à l'aide de l'équation ci-dessous [11] la longueur n de la séquence de vecteurs aléatoires à appliquer à la S-box afin d'être sûr avec un niveau de confiance choisi d'avoir les 203 vecteurs testant.

$$P[X \leq n] = 1 - \sum_{i=1}^k (-1)^{i+1} C_k^i (1-ip)^n$$

où $P[X \leq n]$ est le niveau de confiance choisi (ici fixé à 99%), k est le nombre de vecteurs déterministes cible (ici 203), p est la probabilité d'obtenir un vecteur déterministe (ici $1/2^8$) et n est le nombre de vecteurs aléatoires qu'il faut appliquer pour tester une S-box soit :

$$0,99 = 1 - \sum_{i=1}^{203} (-1)^{i+1} C_{203}^i \left(1 - i \frac{1}{2^8}\right)^n$$

Le résultat de cette équation est illustré sur la figure 2, le niveau de confiance (0,99) est représenté par le trait épais horizontal. Le test d'une S-box nécessite donc l'application de 2534 vecteurs aléatoires pour être sûr à 99% d'avoir les 203 vecteurs déterministes.

Etant donné que dans notre implantation de l'AES les S-boxes sont testées en parallèle, l'ensemble de l'opération Sub Bytes est testée avec 2534 vecteurs aléatoires.

L'expérience a été réalisée avec d'autres implantations des S-boxes, dans tous les cas traités, la

valeur de k (vecteurs déterministes minimal requis) s'étend de 170 à 256 vecteurs déterministes, ce qui donne une longueur de séquence de vecteurs aléatoires allant de 2400 à 2600 vecteurs. La borne supérieure est donc de 2600 vecteurs aléatoires pour tester les S-boxes d'un AES quelque soit leurs implantations.

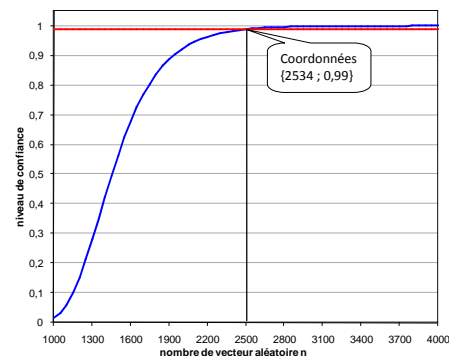


Figure 2 : Longueur de la séquence aléatoire

Concernant, les autres opérations de rondes :

- Shift Rows est implanté uniquement avec des fils, donc pour tester cette opération chacun des fils doit être mis une fois à 0 et une fois à 1 (en considérant comme modèle de faute le modèle de collage). Les vecteurs sortant des S-boxes remplissent cette condition (2600 vecteurs aléatoires sont appliqués aux S-boxes qui sont des opérations de bijection).

- L'opération Mix Columns et Add Round Key sont toutes deux implantés sous forme d'arbre de ou-exclusif. Les vecteurs sortant de l'opération Shift Rows pourront facilement les tester.

En résumé d'un point de vue théorique l'ensemble des opérations de la ronde AES seront testés quand 2600 cycles de rondes ou moins seront effectués.

5. Résultats de l'auto-test d'un AES

Afin de valider notre approche théorique, nous avons fait de la simulation de fautes sur notre cœur AES en mode d'auto-test. Pour les simulations, nous avons utilisé l'outil Tetramax de Synopsys [12].

Nous avons pour cela étudié premièrement une implantation de ce que nous appellerons le chemin de ronde voir figure 3a et dans un deuxième temps une implantation complète de l'AES voir figure 3b.

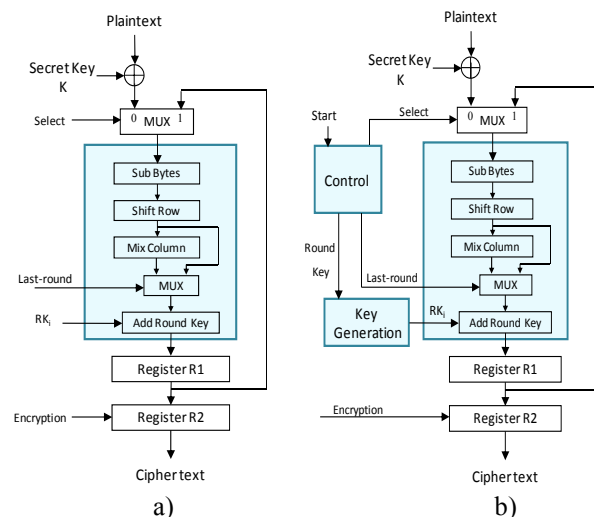


Figure 3 : Les implantations étudiées

La procédure de simulation est la suivante : nous appliquons un texte en clair et une clef secrète quelconque, puis nous réalisons 2534 cycles de rondes (longueur minimale de la séquence de test) et nous observons la sortie (Cipher text). Cette sortie est comparée à celle attendue. Pour la première implantation, les fautes considérées sont uniquement celles de la ronde tandis que dans la deuxième implantation les fautes considérées sont les fautes de la ronde, du générateur de clef et du contrôleur. Les fautes sur le texte en clair, la clef secrète et le cipher text ne sont pas considérées, étant donné qu'il ne sera pas possible d'avoir pour chacune des lignes le test du collage à 1 et du collage à 0 car chaque ligne prendra une seule valeur.

Nous avons appliqué 3 couples différents {texte en clair P, clef secrète K} à chacune des implantations étudiées et pour chacune nous avons effectués 2534 cycles de rondes. Les deux implantations sont entièrement testées, l'ensemble des fautes testables est détectée en sortie. En résumé, un AES réalisant 2534 cycles de rondes est entièrement testé.

Nous avons voulu évaluer de façon expérimentale la taille minimale de la séquence de test pour le chemin de ronde. Le tableau 1 présente les résultats obtenus. La première colonne contient le nombre de cycles de rondes réalisés, les 3 suivantes contiennent les résultats en termes de fautes non testées (ligne supérieure) et le taux de couverture (ligne inférieure) respectivement pour chacun des couples (P, K) étudiés. Par exemple, pour le couple (P1, K1), si 100 cycles de rondes sont réalisés, 8974 fautes sur les 56052 fautes à tester ne sont pas détectées et le taux de couverture est de 83,99%.

En utilisant une approche dichotomique, nous avons obtenu le nombre de rondes exact à réaliser pour tester toutes les fautes. Suivant le couple (P1, K1), (P2, K2) ou (P3, K3) respectivement 2056, 2005 ou 1979 cycles de ronde seront nécessaires.

En résumé, le calcul théorique sur la longueur minimale de la séquence de test est vérifié de façon pratique. Etant donnée les différentes implantations logiques possibles il est préférable d'appliquer environ 2600 cycles de rondes sur un AES pour être sûr d'avoir testé l'ensemble des fautes.

Nombre de cycles de rondes	Couple (P1, K1)	Couple (P2, K2)	Couple (P3, K3)
100	8974	8858	8856
	83,99%	84,20%	84,20%
1000	113	105	112
	99,80%	99,81%	99,80%
2000	1	2	0
	100%	100%	100%
2200	0	0	-
	100%	100%	-

Tableau 1 : Evolution du TC suivant le nombre cycles de rondes

Pour ce qui concerne la surface additionnelle induite par les modifications apportées au cœur AES original, elles représentent au total 3,31% de la surface initiale en comptant non seulement les modifications nécessaire au

mode auto-test présenté ici mais aussi les modes de fonctionnement générateur de vecteurs de test aléatoires et compresseur de signature pour le test d'autres cœurs du système. En comparaison, un registre BILBO [1] représenterait une augmentation de 5,60% pour implanter les mêmes fonctions.

6. Conclusions

Dans le contexte des circuits sécurisé, une approche de test interne (BIST) apparaît comme la solution de test car elle ne fournit pas d'accès externe aux données. Cependant cette approche est coûteuse en termes de surface additionnelle implantation d'un générateur de vecteur de test, d'un analyseur de signature et de la logique de contrôle du mode de test.

Dans ce papier, la solution présentée permet de réduire le surcoût matériel d'une approche BIST, en ajoutant au cœur de chiffrement un mode d'auto-test. L'efficacité de cette solution de test interne aléatoire a été démontrée pour tester l'AES. De plus en ayant une approche pessimiste (une seule combinaison de 203 vecteurs déterministe) et en se fixant un niveau de confiance élevé 99%, nous avons montré que l'AES peut s'auto-tester s'il réalise 2534 cycles de rondes successifs.

Références

- [1] Editeur C. Landrault, "Test de circuits et de systèmes intégrés", HERMES Lavoisier série EGEM, 2004, ISBN 2-7462-0864-4.
- [2] B. Yang, K. Wu, R. Karri, "Scan Based Side Channel Attack on Dedicated Hardware Implementations of Data Encryption Standard", Proc. International Test Conference, 2004, pp 339-344.
- [3] B. Yang, K. Wu, R. Karri, "Secure Scan: A Design-for-Test Architecture for Crypto Chips", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Oct. 2006, pp 2287-2293.
- [4] D. Hély, "TESTABILITY OF SECURE ICS", thèse au LIRMM, soutenue le 2 décembre 2005, Université Montpellier II
- [5] V. Ocheretnij, G. Kouznetsov, R. Karri, M. Gössel, "On-line Error Detection and BIST for the AES Encryption Algorithm with Different S-Box Implementations", IEEE International On-Line Testing Symposium, 2005, pp 141-146.
- [6] G. Di natale, M.-L. Flottes, B. Rouzeyre, "A Dependable Parallel Architecture for SBoxes", Reconfigurable Communication-Centric SoCs ReCoSoc'07, 2007
- [7] Bertoni, et al., "Error Analysis and Detection Procedures for a Hardware Implementation of the AES", IEEE Trans. Computers, Apr. 2003
- [8] M. Doucier, M.-L. Flottes, B. Rouzeyre, "AES-based BIST: self-test, test pattern generation and signature analysis", International Symposium on Electronic Design, Test and Applications, 2008, sous presse.
- [9] FIPS 197, Advanced Encryption Standard (AES), November 2001.
- [10] A. Schubert, W. Anheier, "On Random Pattern Testability of Cryptographic VLSI Cores", Journal of Electronic Testing: Theory and Applications archive, June 2000, Volume 16, Issue 3, pp 185-192.
- [11] S. Shioda, "Some upper and lower bounds on the coupon collector problem", Journal of Computational and Applied Mathematics, March 2007, Volume 200, Issue 1, pp 154-167.
- [12] www.synopsys.com/products/test/tetramax_ds.html