

Amélioration de la sécurité des circuits intégrés par codage de l'information

Julien Francq¹, Jean-Baptiste Rigaud¹, Pascal Manet², Jean-Claude Bajard³, Arnaud Tisserand³

¹ENSMSE, ²CEA-LETI

^{1,2}Centre Microélectronique de Provence Georges Charpak, Laboratoire SESAM^o

Avenue des Anémones - Quartier Saint-Pierre, 13120 GARDANNE

³LIRMM, 161 rue Ada, 34392 MONTPELLIER Cedex 05

Email : ¹{nom@emse.fr}, ²{prénom.nom@cea.fr}, ³{prénom.nom@lirmm.fr}

Résumé

La carte à puce, comme tout autre processeur de traitement d'informations confidentielles peut faire l'objet de manipulations frauduleuses, appelées communément attaques. Cet article décrit les algorithmes cryptographiques les plus couramment embarqués dans ces circuits, puis les principales attaques répertoriées sur de telles structures. Certaines parades existantes à ces attaques sont également abordées, avant d'en évoquer d'autres en cours de développement.

1. Introduction

La plupart des circuits intégrés dédiés à des applications de sécurité disposent de routines cryptographiques afin de garantir suivant les cas la confidentialité, l'intégrité ou l'authenticité des informations échangées avec le monde extérieur [1]. Elles peuvent aussi permettre de vérifier l'identité des parties communicantes.

De plus en plus d'attaques sophistiquées voient le jour afin de porter atteinte à l'intégrité de ces circuits [2]. Ces attaques ont pour but d'extraire tout ou partie des données jugées sensibles, qu'elles soient stockées ou produites par le circuit à un instant donné. A l'heure actuelle, les attaquants disposent d'un important panel d'attaques théoriques, qu'elles soient actives ou passives, auxquelles les concepteurs doivent s'adapter en insérant dans leurs circuits des parades, également appelées contre-mesures. Ces dernières doivent impacter au minimum les contraintes de notre système en termes de temps, de consommation et surtout de surface. L'idéal pour les concepteurs serait d'imaginer des attaques novatrices dans le but d'insérer dans le système de nouvelles contre-mesures adaptées, afin d'avoir un temps d'avance sur d'éventuels attaquants.

Enfin, il est nécessaire de maîtriser parfaitement l'état de l'art dans le domaine des contre-mesures afin de durcir ces circuits vis-à-vis des attaques actuelles et le cas échéant, concevoir constamment de nouvelles parades.

La section 2 est consacrée à la présentation des algorithmes les plus utilisés pour réaliser de la cryptographie à clé publique ou privée. La section 3

détaille les principales attaques connues sur ces algorithmes. Les principales contre-mesures existantes vis-à-vis de ces attaques, les futurs axes de recherche ainsi que la méthodologie de validation de ces parades seront présentés dans la section 4 avant de conclure.

2. Algorithmes cryptographiques

Afin d'assurer la confidentialité des messages traités, ceux-ci peuvent être chiffrés en utilisant des algorithmes paramétrés par des clés. Pour cela, deux types de cryptographie peuvent être utilisés : la cryptographie symétrique (ou à clé privée) et la cryptographie asymétrique (ou à clé publique) [1].

2.1 Cryptographie symétrique

Dans cette sous-section, il ne sera évoqué que les protocoles de chiffrement par blocs, dans lesquels le texte clair est divisé en blocs de taille fixe. Ensuite, un seul bloc est chiffré à la fois. Ce protocole est à différencier du chiffrement par flot dans lequel le texte clair est chiffré bit à bit [1].

Dans la deuxième partie des années 70, le premier standard de chiffrement symétrique apparaissait : le DES (*Data Encryption Standard*) [3]. Bien que sûr à l'époque d'un point de vue mathématique, l'utilisation de cet algorithme est limitée à cause de la taille de sa clé secrète. En effet, une recherche exhaustive de cette dernière conduit à tester 2^{56} clés candidates, ce qui est rendu possible par l'augmentation de la puissance de calcul des processeurs. Il est donc apparu nécessaire d'adopter rapidement un nouveau standard de cryptographie symétrique avec une taille de clé augmentée. C'est ainsi que l'algorithme AES (*Advanced Encryption Standard*) a été adopté comme standard en 2001 [4].

Le chiffrement AES s'effectue sur des blocs de données de 128 bits, en utilisant des clés de chiffrement de longueur égale à 128, 192 ou 256 bits. Un bloc peut être représenté sous la forme d'une matrice 4x4 d'octets, également appelée *state*.

On peut décomposer le chiffrement AES en deux suites d'opérations se déroulant en parallèle : l'*encryption process*, qui se charge de l'exécution des opérations s'effectuant sur le *state* et le *Key Schedule*, qui calcule les différentes clés de rondes à partir de la clé de chiffrement

^o Ce travail fait l'objet d'une thèse à financement Fonds Social Européen (FSE) dans le cadre du projet BTRS (Briques Technologiques pour le Renforcement de la Sécurité), dont les partenaires industriels sont Gemalto et SPS.

^o Le laboratoire SESAM est une équipe mixte CEA-LETI / Ecole des Mines de Saint-Étienne basée sur le site Georges Charpak (Gardanne).

initiale. Le chiffrement est une suite de transformations mathématiques appelée ronde, itérée N_r fois, où N_r dépend de la longueur de la clé. Une ronde est composée de quatre transformations élémentaires : *SubBytes*, *ShiftRows*, *MixColumns* et *AddRoundKey*. La première transformation est une substitution non-linéaire, les deux suivantes forment une couche de diffusion, et la dernière représente l'ajout d'informations secrètes à chaque ronde.

D'autres algorithmes symétriques existent (FEAL, IDEA, SAFER, RC5, etc. [1]), mais le DES et l'AES sont les plus utilisés car ce sont les seuls standard de chiffrement symétrique.

Les algorithmes symétriques possèdent des performances intéressantes pour le chiffrement des données, mais ils ne sont pas utilisables dans certains protocoles, comme par exemple ceux effectuant la signature de données échangées. L'utilisation d'algorithmes asymétriques peut être une solution.

2.2 Cryptographie asymétrique

En 1976, une nouvelle forme de cryptographie apparaissait : la cryptographie à clé publique, avec l'algorithme RSA.

Ce type de cryptographie repose sur l'existence de deux clés, l'une publique et l'autre privée. Il peut être utilisable notamment pour garantir l'authentification des parties communicantes ou l'intégrité des données échangées. Dans le cadre de l'authentification, l'expéditeur chiffre ses données avec sa clé privée : ainsi, n'importe quel possesseur de la clé publique peut vérifier son identité. Dans le cadre de l'intégrité des données, n'importe quel expéditeur pourra chiffrer un texte clair avec la clé publique du destinataire, et seul celui-ci pourra le déchiffrer avec sa clé privée.

Utilisant ces concepts, l'algorithme RSA repose sur l'existence de deux couples de clés : l'un est public (n, e) et l'autre privé (n, d) . Dans ce qui suit, la génération de ces couples de clés n'est pas détaillée : il n'est expliqué que le chiffrement RSA en lui-même.

Dans le cadre du chiffrement de données, l'expéditeur du texte clair doit effectuer une exponentiation modulaire : si P est le texte clair et C le texte chiffré, il doit calculer $C = P^e \bmod n$. Pour déchiffrer, le destinataire du message doit calculer $P = C^d \bmod n$. Le nombre n est le produit de deux grands nombres premiers p et q qui sont eux-mêmes utilisés pour calculer l'exposant public e et surtout l'exposant privé d .

La sécurité de cet algorithme repose sur la factorisation des grands nombres. En effet, n est dans le domaine public, mais il n'existe pas actuellement d'algorithmes efficaces permettant de retrouver les facteurs premiers p et q de n .

Pour garantir un haut niveau de sécurité, on doit pouvoir choisir une taille suffisamment grande pour n . Avec la puissance de calcul actuelle des processeurs, on estime que la taille de n doit être au minimum de 2048 bits.

Une structure possible pour le calcul de l'exponentiation modulaire utilise des multiplieurs de Montgomery [5]. D'un point de vue matériel, celle-ci est reconnue comme étant une des plus rapides et des plus

adaptées pour des calculs impliquant des opérandes de grande taille.

Les algorithmes cryptographiques présentés dans cette section sont considérés comme sûrs d'un point de vue mathématique, mais leur réalisation matérielle les rend sensibles à des attaques dont certaines sont présentées en section 3.

3. Principales attaques connues

Dans cette partie, il n'a été évoqué qu'un certain nombre d'attaques. Ce n'est pas un descriptif exhaustif, et il ne sera question que d'attaques connues sur l'AES.

Les attaques connues dans la littérature peuvent être classées en deux catégories, selon qu'elles sont passives ou actives. Les attaques passives étant très étudiées depuis 1998, cette étude se concentre sur les attaques actives.

Ce type d'attaque consiste à modifier l'environnement du circuit (température, exposition à des rayonnements UV, X ou visibles, modification de la fréquence d'horloge, etc.) de telle sorte que le fonctionnement de celui-ci s'en trouve altéré [6]. Il a été montré dans la littérature que cette attaque dite par « injection de fautes » pendant l'exécution d'un algorithme est un moyen efficace pour récupérer la clé de chiffrement d'un cryptosystème [7] ou pour contourner des protections.

Un premier type d'attaque en fautes est l'attaque DFA (*Differential Fault Analysis*) qui utilise les différences entre des exécutions normales et fautes d'un algorithme cryptographique pour obtenir des informations sur la clé de chiffrement. Plusieurs attaques DFA ont été répertoriées, tout d'abord sur le DES puis sur l'AES.

On peut citer comme premier exemple d'attaque DFA l'attaque de Giraud sur un bit [8] : le principe de cette attaque est de fausser un seul bit d'un octet juste avant l'exécution du dernier *SubBytes* (c'est-à-dire en entrée du *SubBytes* de la 10^{ème} ronde ou du *AddRoundKey* de la 9^{ème} ronde). En répétant la même procédure pour tous les octets, cette attaque permet de retrouver tous les octets de clé de la 10^{ème} ronde ; la clé de chiffrement est donc calculable entièrement en exécutant l'inverse de la fonction *Key Schedule*. Le principal avantage de cette attaque est qu'il n'est pas nécessaire de connaître l'octet ou le bit fauté. Cette attaque peut être réalisable en mettant en œuvre des moyens matériels importants ainsi qu'une compétence certaine de l'attaquant. Dans le même article, Giraud présente une seconde attaque qui semble techniquement moins exigeante puisqu'elle nécessite de fauter un octet.

Un autre exemple d'attaque DFA sur l'AES est l'attaque de Piret et de Quisquater [9]. Dans un premier temps, on doit établir une liste qui est celle de toutes les fautes possibles sur un octet pour une colonne à l'entrée du *MixColumns* de la 9^{ème} ronde, soit 1020 fautes possibles (une colonne est constituée de 4 octets et il existe 255 possibilités de valeurs de fautes pour chaque octet). On calcule ensuite le *MixColumns* de toutes ces fautes possibles. On raisonne sur des différences, donc l'ajout de la clé de la 9^{ème} ronde est inutile. On dispose

ainsi d'une liste de toutes les différences qu'on peut observer en entrée du *SubBytes* de la 10^{ème} ronde pour une colonne et pour une faute qui se serait produite entre les *MixColumns* de la 8^{ème} et de la 9^{ème} ronde. La deuxième étape de cette attaque consiste à parcourir l'algorithme AES dans le sens du déchiffrement, avec comme point de départ le texte chiffré vrai et le texte chiffré erroné. Pour ces deux textes, on commence par sélectionner les deux premiers octets d'une colonne, puis on effectue l'opération inverse du *AddRoundKey* de la 10^{ème} ronde avec toutes les possibilités de clé partielles (2^{16} candidates) sur chacun d'entre eux. On fait ensuite passer l'ensemble de ces résultats dans la fonction inverse du *SubBytes* et on calcule la différence entre les résultats intermédiaires vrai et erroné pour chaque hypothèse de clé. On vérifie si cette différence existe dans la table et, si c'est le cas, on conserve la clé partielle candidate. Pour déterminer entièrement la colonne de la matrice représentant la clé, on sélectionne le 3^{ème} octet des textes chiffrés vrai et erronés, puis on ajoute le troisième octet de clé (ce qui donne un nombre de possibilités égal à $256 \times$ nombre de clés partielles candidates trouvées en 1^{ère} phase), on réitère tout le processus, puis on termine le traitement de la colonne. Ainsi, avec seulement 2 couples de chiffrés corrects et faux, on retrouve toute la colonne de clé dans 97% des cas. En répétant cette démarche sur chaque colonne, on trouve toute la clé. En injectant la faute lors de la ronde précédente, on peut trouver toute la clé avec deux couples de chiffrés corrects et faux.

D'autres attaques DFA utilisent les mêmes idées de base que les précédentes ([10], [11]).

Un deuxième type d'attaque en fautes utilisable est l'attaque dite en *safe-error*. Dans ce type d'attaque, il suffit seulement de déterminer si la ou les fautes insérées amènent des différences sur les textes chiffrés, tandis que les attaques DFA exploitent la différence elle-même ([12] pour le RSA, [13] pour l'AES).

Il existe également des attaques sur les machines d'état contrôlant le déroulement de l'exécution des fonctions cryptographiques [14] et non sur le chemin de données. En réduisant le nombre de rondes d'un algorithme, la clé peut être récupérée plus rapidement.

Les concepteurs de circuits sécurisés doivent être sensibilisés à toutes les attaques existantes afin d'implémenter des contre-mesures adaptées.

4. Contre-mesures

Cette section se concentre principalement sur les contre-mesures aux attaques actives présentées dans la section 3. L'apport de la logique asynchrone pour la conception de circuits sécurisés ne sera pas développé dans cet article [15], car cette thématique n'est pas l'objet de notre étude.

4.1 Quelques contre-mesures existantes

Cette sous-section énumère trois méthodes de durcissement de l'AES vis-à-vis des attaques en fautes ainsi qu'une méthode de durcissement pour le RSA.

Pour l'AES, Guido Bertoni [16] propose une contre-mesure basée sur l'utilisation d'un bit de parité par octet

et d'une matrice d'erreur. Ce type de contre-mesure est intéressant car il induit l'ajout d'une quantité réduite de portes logiques. Son principal inconvénient est que toutes les fautes inversant un nombre pair de bits sur un octet ne seront pas détectées.

Une autre contre-mesure proposée pour l'AES induit un surcoût plus important [17]. Le principe est de déchiffrer chaque groupe de transformations en parallèle du chiffrement et de comparer la sortie du déchiffrement avec l'entrée avant le chiffrement. Il y a dans ce cas une double perte en terme de surface et de temps : un surcoût en surface de l'ordre de la duplication puisque chaque brique doit être décodée, et un surcoût en temps puisque chaque élément doit être mémorisé pour être comparé avec le même élément codé et décodé. Si l'on dispose à la fois du chiffrement et du déchiffrement sur le même circuit, cette contre-mesure induit un surcoût modéré, par contre elle augmente considérablement le temps d'exécution.

Une autre contre-mesure possible se base sur une triplication du chemin de données avec décision prise par vote majoritaire [18]. En raison du surcoût induit en surface et en temps, l'insertion matérielle d'une telle contre-mesure est exclue dans le contexte de la carte à puce.

L'insertion de contre-mesures pour le RSA est aussi possible. Par exemple, l'intégrité de calcul des multiplications modulaires peut être validée avec une très grande probabilité grâce à la preuve par $2^n - 1$, qui reprend l'idée de la preuve par 9 [19].

De nouvelles contre-mesures peuvent être imaginées en s'inspirant de celles décrites dans cette sous-section.

4.2 Futurs axes de recherche

Il est envisagé trois axes principaux de recherche pour cette étude.

Tout d'abord, dans le but de protéger les cryptosystèmes contre certaines perturbations provoquées intentionnellement ou par son environnement immédiat, l'insertion de blocs détecteurs ou correcteurs d'erreurs dans la structure de ces circuits semble être une approche prometteuse. Cependant, il est à noter que les techniques de codage ne sont actuellement pas utilisées pour sécuriser les algorithmes de cryptographie à clef privée vis-à-vis des attaques en fautes. En effet, l'insertion de codes évolués dans de telles structures revient à résoudre des problèmes mathématiques difficiles. Par exemple, pour l'AES, les difficultés résultent de la non-linéarité de la transformation *SubBytes*. Une fois le problème mathématique résolu, la difficulté repose sur l'implémentation matérielle de ce code ayant à la fois la plus grande capacité de correction possible et des caractéristiques en temps et en surface intéressantes.

Ensuite, pour les deux types de cryptographie, des bibliothèques de cellules résistantes aux fautes effectuant des opérations élémentaires dans les corps finis pourront être développées (addition, multiplication, inversion).

Enfin, un gain de sécurité peut être espéré grâce à un changement dynamique du format du chemin de données pendant l'exécution de calculs cryptographiques intermédiaires.

L'implémentation de ces nouvelles contre-mesures censées durcir notre système vis-à-vis des attaques en fautes ne devra pas accentuer la faiblesse de notre système vis-à-vis des attaques passives.

4.3 Validation matérielle de ces contre-mesures

Pour les concepteurs de circuits sécurisés, il est intéressant de posséder un outil simulant le comportement d'un circuit sécurisé lorsque des fautes y sont injectées [20], et pouvant également permettre d'évaluer l'efficacité des contre-mesures à certaines attaques. Une telle plate-forme émulant une carte à puce simplifiée a déjà été développée et validée sur FPGA. Cette plate-forme embarque un bloc matériel d'injection de fautes [21] destiné à la validation de contre-mesures.

5. Conclusions

Dans cet article, les algorithmes les plus utilisés en cryptographie à clé privée et à clé publique ont été décrits. Les attaques les plus efficaces sur ces algorithmes, qu'elles soient actives ou passives ont également été énumérées. Des parades aux attaques en fautes existantes ont été décrites, et, dans un souci purement prospectif, d'autres concepts de contre-mesures ont été évoqués.

La prochaine étape de cette étude pourra être l'insertion d'un code correcteur dont la construction mathématique est relativement simple (par exemple un codeur de Hamming) dans la structure d'un cryptoprocésseur à clé privée comme l'AES. Ceci nous permettrait de durcir notre système contre l'attaque de Giraud sur un bit et contre les attaques en *safe-error*.

Références

[1] A.J. Menezes, P.C. van Oorschot, S.C. Vanstone. "Handbook of Applied Cryptography". CRC Press, Boca Raton, 1997.

[2] H. Bar-El, H. Choukri, D. Naccache, M. Tunstall and C. Whelan. "The sorcerer's apprentice guide to fault attacks". In First Workshop on Fault Detection and Tolerance in Cryptography – FDTC 2004, Florence, Italy.

[3] NIST. "Announcing the Data Encryption Standard (DES)". FIPS PUB 46-3 (Federal Information Processing Standards Publication), October 25, 1999.

[4] NIST. "Announcing the Advanced Encryption Standard (AES)". Federal Information Processing Standards Publication, n.197, November 26, 2001.

[5] P. Montgomery. "Modular Multiplication Without Trial Division". Mathematics of Computation, vol. 44, 1995.

[6] R. J. L. Dan Boneh, R. Demillo. "On the Importance of Checking Cryptographic Protocols for Faults", Advances in Cryptology, Proc. Eurocrypt, pp. 37-51, 1997.

[7] E. Biham and A. Shamir. "Differential Fault Analysis of secret key cryptosystems". In B.S. Kaliski Jr., editor, Advances in Cryptology – CRYPTO, vol. 1294 of Lecture Notes in Computer Science, pp. 513–525. Springer, 1997.

[8] C. Giraud. "DFA on AES". In H. Dobbertin, V. Rijmen, and A. Sowa, editors, Advanced Encryption Standard – AES, vol. 3373 of Lecture Notes in Computer Science, pp. 27–41. Springer, 2005.

[9] G. Piret and J.-J. Quisquater. "A differential fault attack technique against SPN structures, with application to the AES and Khazad". In C.D. Walter, editor, Cryptographic Hardware and Embedded Systems – CHES, vol. 2779 of Lecture Notes in Computer Science, pp. 77–88. Springer, 2003.

[10] P. Dusart, G. Letourneux, and O. Vivilo. "Differential fault analysis on A.E.S.". In J. Zhou, M. Yung, and Y. Han, editors, Applied Cryptography and Network Security – ACNS, vol. 2846 of Lecture Notes in Computer Science, pp. 293–306. Springer, 2003.

[11] A. Moradi, M.T. Manzuri Shalmani, M. Salmasizadeh. "A Generalized Method of Differential Fault Attack Against AES". In L. Goubin and M. Matsui, editors, Cryptographic Hardware and Embedded Systems – CHES, vol. 4249 of Lecture Notes in Computer Science, pp. 91–100. Springer, 2006.

[12] S.-M. Yen and M. Joye. "Checking before output may not be enough against fault-based cryptanalysis". IEEE Transactions on Computers, 49(9):967–970, 2000.

[13] J. Blömer and J.-P. Seifert. "Fault based cryptanalysis of the Advanced Encryption Standard". In R.N. Wright, editor, Financial Cryptography (FC), vol. 2742 of Lecture Notes in Computer Science, pp. 162–181. Springer, 2003.

[14] H. Choukri and M. Tunstall. "Round reduction using faults". In Second Workshop on Fault Detection and Tolerance in Cryptography (FDTC), Edinburgh, UK, September 2005.

[15] Yannick Monnet, Marc Renaudin, Régis Leveugle. "Designing Resistant Circuits against Malicious Faults Injection Using Asynchronous Logic". IEEE Trans. Computers 55(9): 1104-1115, 2006.

[16] G. Bertoni, L. Breveglieri, I. Koren, P. Maistri, "An Efficient Hardware-Based Fault Diagnosis Scheme for AES: Performance and Cost", in the proc. of DFT, Cannes, France, October 2004.

[17] R. Karri, K. Wu, P. Mishra, Y. Kim, "Concurrent Error Detection Schemes for Fault-Based Side-Channel Cryptanalysis of Symmetric Block Ciphers", in IEEE Transactions on Computer-Aided Design, vol. 21, num. 12, pp. 1509-1517, December 2002.

[18] F. Lima, L. Carro, and R. Reis. "Designing fault tolerant systems into sram-based fpgas". In Design Automation Conference (DAC): Proc. pp. 650–655, New York, NY, USA, 2003. ACM Press.

[19] S. Guilley and P. Hoogvorst, "Proof by 2^m-1 : a low-cost method to check arithmetic computations", SEC 2005, May 30 - June 1, 2005, Makuhari-Messe, Chiba, Japan.

[20] O. Faurax, L. Freund, A. Tria, T. Muntean and F. Bancel. "A generic method for fault injection in circuits", In IWSOC 2006: Proc. of the 6th IEEE International Workshop on System-on-Chip for Real-Time Applications, pp. 211–214.

[21] J. Francq, P. Manet and J.-B. Rigaud. "Material Emulation of Faults On Cryptoprocessors". In Proc. of Sophia Antipolis forum of MicroElectronics (SAME), France, 2006.