

NNT : 2011 EMSE 0636

THÈSE

présentée par

Amir-Pasha MIRBAHA

pour obtenir le grade de
Docteur de l'École Nationale Supérieure des Mines de Saint-Étienne

Spécialité : Microélectronique

ÉTUDE DE LA VULNÉRABILITÉ DES CIRCUITS CRYPTOGRAPHIQUES À L'INJECTION DE FAUTES PAR LASER

soutenue à Gardanne, le 20 décembre 2011

Membres du jury

Président :	Régis LEVEUGLE	Professeur, TIMA, Grenoble
Rapporteurs :	Philippe MAURINE	Maître de conférences, LIRMM, Montpellier
	Jean-Jacques QUISQUATER	Professeur, UCL, Louvain-la-Neuve
Directeurs de thèse :	Assia TRIA	Directrice de laboratoire, CEA-LETI, Gardanne
	David NACCACHE	Professeur, ENS, Paris
Encadrant de thèse :	Jean-Max DUTERTRE	Maître-assistant, ENSM SE, Gardanne

Spécialités doctorales :

SCIENCES ET GENIE DES MATERIAUX
 MECANIQUE ET INGENIERIE
 GENIE DES PROCEDES
 SCIENCES DE LA TERRE
 SCIENCES ET GENIE DE L'ENVIRONNEMENT
 MATHEMATIQUES APPLIQUEES
 INFORMATIQUE
 IMAGE, VISION, SIGNAL
 GENIE INDUSTRIEL
 MICROELECTRONIQUE

Responsables :

K. Wolski Directeur de recherche
 S. Drapier, professeur
 F. Gruy, Maître de recherche
 B. Guy, Directeur de recherche
 D. Graillot, Directeur de recherche
 O. Roustant, Maître-assistant
 O. Boissier, Professeur
 J.C. Pinoli, Professeur
 A. Dolgui, Professeur
 Ph. Collot, Professeur

EMSE : Enseignants-chercheurs et chercheurs autorisés à diriger des thèses de doctorat (titulaires d'un doctorat d'État ou d'une HDR)

AVRIL	Stéphane	MA	Mécanique & Ingénierie	CIS
BATTON-HUBERT	Mireille	MA	Sciences & Génie de l'Environnement	Fayol
BENABEN	Patrick	PR 1	Sciences & Génie des Matériaux	CMP
BERNACHE-ASSOLLANT	Didier	PR 0	Génie des Procédés	CIS
BIGOT	Jean-Pierre	MR	Génie des Procédés	SPIN
BILAL	Essaïd	DR	Sciences de la Terre	SPIN
BOISSIER	Olivier	PR 1	Informatique	Fayol
BORBELY	Andras	MR	Sciences et Génie des Matériaux	SMS
BOUCHER	Xavier	MA	Génie Industriel	Fayol
BRODHAG	Christian	DR	Sciences & Génie de l'Environnement	Fayol
BURLAT	Patrick	PR 2	Génie industriel	Fayol
COLLOT	Philippe	PR 1	Microélectronique	CMP
COURNIL	Michel	PR 0	Génie des Procédés	SPIN
DARRIEULAT	Michel	IGM	Sciences & Génie des Matériaux	SMS
DAUZERE-PERES	Stéphane	PR 1	Génie industriel	CMP
DEBAYLE	Johan	CR	Image, Vision, Signal	CIS
DELAFOSSSE	David	PR1	Sciences & Génie des Matériaux	SMS
DESRAYAUD	Christophe	MA	Mécanique & Ingénierie	SMS
DOLGUI	Alexandre	PR 1	Génie Industriel	Fayol
DRAPIER	Sylvain	PR 2	Sciences & Génie des Matériaux	SMS
FEILLET	Dominique	PR 2	Génie Industriel	CMP
FOREST	Bernard	PR 1	Sciences & Génie des Matériaux	CIS
FORMISYN	Pascal	PR 1	Sciences & Génie de l'Environnement	Fayol
FRACZKIEWICZ	Anna	DR	Sciences & Génie des Matériaux	SMS
GARCIA	Daniel	MR	Sciences de la terre	SPIN
GIRARDOT	Jean-Jacques	MR	Informatique	Fayol
GOEURLOT	Dominique	MR	Sciences & Génie des Matériaux	SMS
GRAILLOT	Didier	DR	Sciences & Génie de l'Environnement	Fayol
GROSSEAU	Philippe	MR	Génie des Procédés	SPIN
GRUY	Frédéric	MR	Génie des Procédés	SPIN
GUY	Bernard	MR	Sciences de la Terre	SPIN
GUYONNET	René	DR	Génie des Procédés	SPIN
HAN	Woo-Suck	CR		SMS
HERRI	Jean-Michel	PR 2	Génie des Procédés	SPIN
INAL	Karim	PR 2	Microélectronique	CMP
KLOCKER	Helmut	DR	Sciences & Génie des Matériaux	SMS
LAFOREST	Valérie	CR	Sciences & Génie de l'Environnement	Fayol
LERICHE	Rodolphe	CR CNRS	Mécanique et Ingénierie	SMS
LI	Jean-Michel	EC (CCI MP)	Microélectronique	CMP
MALLIARAS	George Grégory	PR 1	Microélectronique	CMP
MOLIMARD	Jérôme	PR2	Mécanique et Ingénierie	SMS
MONTHEILLET	Frank	DR 1 CNRS	Sciences & Génie des Matériaux	SMS
PERIER-CAMBY	Laurent	PR 2	Génie des Procédés	SPIN
PIJOLAT	Christophe	PR 1	Génie des Procédés	SPIN
PIJOLAT	Michèle	PR 1	Génie des Procédés	SPIN
PINOLI	Jean-Charles	PR 0	Image, Vision, Signal	CIS
ROUSTANT	Olivier	MA		Fayol
STOLARZ	Jacques	CR	Sciences & Génie des Matériaux	SMS
SZAFNICKI	Konrad	MR	Sciences & Génie de l'Environnement	Fayol
TRIA	Assia		Microélectronique	CMP
VALDIVIESO	François	MA	Sciences & Génie des Matériaux	SMS
VIRICELLE	Jean-Paul	MR	Génie des procédés	SPIN
WOLSKI	Krzysztof	DR	Sciences & Génie des Matériaux	CIS
XIE	Xiaolan	PR 1	Génie industriel	CMP

ENISE : Enseignants-chercheurs et chercheurs autorisés à diriger des thèses de doctorat (titulaires d'un doctorat d'État ou d'une HDR)

FORTUNIER	Roland	PR	Sciences et Génie des matériaux	ENISE
BERGHEAU	Jean-Michel	PU	Mécanique et Ingénierie	ENISE
DUBUJET	Philippe	PU	Mécanique et Ingénierie	ENISE
LYONNET	Patrick	PU	Mécanique et Ingénierie	ENISE
SMUROV	Igor	PU	Mécanique et Ingénierie	ENISE
ZAHOUANI	Hassan	PU	Mécanique et Ingénierie	ENISE
BERTRAND	Philippe	MCF	Génie des procédés	ENISE
HAMDI	Hédi	MCF	Mécanique et Ingénierie	ENISE
KERMOUCHE	Guillaume	MCF	Mécanique et Ingénierie	ENISE
RECH	Joël	MCF	Mécanique et Ingénierie	ENISE
TOSCANO	Rosario	MCF	Mécanique et Ingénierie	ENISE
GUSSAROV Andrey	Andrey	Enseignant contractuel	Génie des procédés	ENISE

Glossaire :

PR 0	Professeur classe exceptionnelle	Ing.	Ingénieur
PR 1	Professeur 1 ^{ère} classe	MCF	Maître de conférences
PR 2	Professeur 2 ^{ème} classe	MR(DR2)	Maître de recherche
PU	Professeur des Universités	CR	Chargé de recherche
MA(MDC)	Maître assistant	EC	Enseignant-chercheur
DR	Directeur de recherche	IGM	Ingénieur général des mines

Centres :

SMS	Sciences des Matériaux et des Structures
SPIN	Sciences des Processus Industriels et Naturels
FAYOL	Institut Henri Fayol
CMP	Centre de Microélectronique de Provence
CIS	Centre Ingénierie et Santé



NNT: 2011 EMSE 0636

Ph.D. Manuscript

to obtain the title of

Doctor

of the École Nationale Supérieure des Mines de Saint-Étienne

Specialty : MICROELECTRONICS

Defended by

Amir-Pasha MIRBAHA

Study of the Vulnerability of Cryptographic Circuits by Laser Fault Injection

prepared at Secure Systems and Architectures, SAS Department

defended on 20 December 2011

Jury:

<i>President:</i>	Régis LEVEUGLE	Professor, TIMA, Grenoble
<i>Reviewers:</i>	Philippe MAURINE	Associate Professor, LIRMM, Montpellier
	Jean-Jacques QUISQUATER	Professor, UCL, Louvain-la-Neuve
<i>Advisors:</i>	Assia TRIA	Head of Laboratory, CEA-LETI, Gardanne
	David NACCACHE	Professor, ENS, Paris
<i>Co-Advisor:</i>	Jean-Max DUTERTRE	Associate Professor, ENSM SE, Gardanne

Acknowledgments

Ce travail de thèse a été réalisé avec une bourse du Ministère français de l'Économie, de l'Industrie et de l'Emploi au Centre Microélectronique de Provence (CMP) à Gardanne (France), au sein de l'équipe mixte CEA-LETI et ENSM-SE Systèmes et Architectures Sécurisés (SAS), dirigée par Madame Assia Tria.

Au bout de cette thèse préparée entre octobre 2008 et décembre 2011, je tiens à adresser ma profonde reconnaissance à :

Madame Assia Tria pour avoir dirigé mes travaux de thèse et m'avoir accordé cette bourse. Je la remercie particulièrement pour la confiance et la disponibilité qu'elle m'a témoignées durant ces trois années.

Professeur David Naccache qui a codirigé cette thèse pour m'avoir orienté vers la cryptographie dès le début de mon doctorat et pour ses conseils, son soutien et son sens de l'exigence.

Jean-Max Dutertre pour m'avoir encadré avec grandes patience et disponibilité pendant la préparation de cette thèse et la relecture de mes communications et mon manuscrit de thèse.

Professeur Jean-Jacques Quisquater pour l'honneur qu'il m'a fait d'accepter d'être rapporteur de cette thèse et l'intérêt qu'il a porté à ce travail.

Philippe Maurine pour l'intérêt qu'il a porté à ce travail de thèse, ses remarques enrichissantes lors des colloques précédents et pour avoir accepté de siéger au jury de ma thèse en qualité de rapporteur.

Professeur Régis Leveugle pour l'honneur qu'il m'a fait d'accepter d'examiner ce travail en qualité d'examinateur et de présider le jury de ma thèse.

Professeur Philippe Collot, Président du Centre Microélectronique de Provence, pour son soutien pendant la préparation de ma thèse.

Aujourd'hui, je garde une belle expérience, des bons souvenirs et des amitiés nouées au fil de ces trois années d'étude. Je souhaite remercier ceux qui m'ont aidé pour mes travaux et mes essais ; notamment :

Michel Agoyan, avec sa grande disponibilité et la qualité de son expertise qui reste pour moi une référence.

Anne-Lise Ribotta qui m'a beaucoup aidé pour la mise en place des essais et la résolution des problèmes. Je la remercie pour sa bonne humeur et sa grande disponibilité à tout moment.

Bruno Robisson, Jean-Baptiste Rigaud et Jacques Fournier pour l'intérêt et la considération qu'ils portaient à mes travaux.

Nicolas Rodriguez, Loïc Lauro, Driss Aboulkassimi, Amine Dehbaoui, Thierry Vaschalde, Cyril Roscian, Marion Verdier et Minh-Huu Nguyen pour le partage de leurs recherches, essais et expériences.

Je tiens également à remercier Pascal Manet, Jérôme Quartana, Serge Seyroles, Sylvain Bouquet, Claude Barral, Claire Pechikoff, Olivier Vallier, Laurent Freund,

Marc Ferro, Florian Praden, Ronan Lashermes, François Poucheret, Guillaume Raymond, Patrick Orsatelli, Nicolas Moro, Hélène Le Boudier, Selma Laabidi, Olivia Bret, Julien Francq, Benjamin Mounier, Rémi Nannini, Loïc Zussa et Alexandre Sarafianos au département SAS pour leur accueil, leur confiance et les expériences professionnelles partagées.

Je remercie également Jacques Legeleux, Jean-Michel Li, Bernard Dhalluin, le Professeur George Malliaras, Dion Khodagholy, Etienne Drahi, Gaëlle Rondeau, Sylvain Blayac, Pauline Sajous, Béatrice Dubois, Sylvain Nolot, Thierry Camilioni, Cyril Calmes, Sabine Salmeron et Lobna Haouari que j'ai plusieurs fois consultés lors de mes travaux pour les agréments de leurs conseils, d'expériences et d'encouragement.

Merci du fond du cœur à Véronique Villaréal pour son ouverture, son soutien et son accompagnement à tout moment dès le jour de mon entretien d'embauche au CMP et jusqu'à l'envoi de cette version finale de mon manuscrit de thèse. Je suis également très reconnaissant à Joëlle Guelon pour sa gentillesse et son soutien.

Un grand merci également à Barbara Bruno pour son accueil chaleureux et ces trois ans et demi d'aides administratives ; ainsi qu'à Anne-Marie Caillet, Valérie Perret, Chrystelle Wojciechowski, Thierry Ricordeau, Serge Burgun, Alice Durieu, Adélaïde Crabit et Michelle Gillet.

Par ailleurs, je remercie Michael Haggerty pour pour la relecture attentive de ce manuscrit et les corrections d'anglais. Je suis également reconnaissant à Gracien Counot, Stéphane Isnard, Axel Seguin, Manon Leoni, Jonathan Alarcon et Florent Bitschy pour leurs soutiens informatiques. Je remercie également tout le reste du personnel du CMP pour leur accueil.

J'exprime mon hommage au Professeur Alain Vautrin, ancien Directeur de l'École doctorale 488 SIS et personnage exceptionnel qui était toujours à l'écoute des doctorants et travaillait sans cesse au perfectionnement de la qualité des services de l'ED-SIS jusqu'à sa triste disparition. Je remercie également le Professeur Christophe Donnet, le Professeur Christophe Desrayaud et les autres élus doctorants pour leurs retours d'expérience ; notamment Alexandre Franquet pour sa forte participation à toutes les activités et Brice Arrazat pour sa collaboration. Merci aussi à Romain Cauchois pour son engagement et son implication de représenter les doctorants du CMP auprès de l'Équipe de Direction (EqDir).

De plus, je suis très reconnaissant à mon professeur de longue date, Keivan Navi à l'Université Shahid Béheshhti qui m'a toujours encouragé et soutenu pour accomplir mes études de doctorat et pour m'avoir orienté vers les sujets de l'arithmétique et du matériel des ordinateurs. Je remercie également Omid Kavehei et Mostafa Rahimi Azghadi pour leurs amitiés et les recherches et les expériences enrichissantes que nous avons effectuées ensemble entre 2005 et 2007 et qui m'ont aidé pour cette nouvelle orientation.

Par ailleurs, je souhaite remercier profondément Martine Dorance, la Conseillère culturelle, scientifique et de coopération de France en Iran, Bernard Paquetteau,

l'Attaché scientifique et technique et son successeur, Dr. Sixte Blanchy ; ainsi que Zohreh Mirbaha, la Responsable du Service des étudiants et Armand Karimi Goudarzi, mon ancien professeur de langue et littérature françaises, au Service de Coopération et d'Action Culturelle de France en Iran pour m'avoir accordé en 2003-2005 leurs soutiens et des bourses d'études en France. Elles m'ont permis de réaliser mon rêve de poursuivre mes études dans les meilleures conditions en France. Je remercie également mes anciens professeurs, Thierry Baffoy à Paris 8 et le Professeur Colette Rolland à Paris 1 pour m'avoir aidé à trouver la formation doctorale adéquate.

Enfin, je souhaite adresser mes sincères remerciements à tous les miens pour m'avoir transmis les valeurs qui donnent de la richesse à l'existence : Je souhaite remercier particulièrement mes parents et ma sœur qui m'entourent et m'ont toujours soutenu par leurs affections. Je présente ma gratitude à ma grand-mère qui m'encourageait pour une réussite professionnelle et qui malheureusement m'a quitté pendant la préparation de cette thèse. Je présente ma gratitude à mon cher ami, Mohamed-Amine Kechiche, toujours présent à mes côtés dans les moments difficiles. Je remercie également Rahim Moazami pour son amitié et sa présence permanente à côté de ma famille pendant cette période. Merci à Shiva Rouholamini de son amitié, de m'avoir aidé pour ma soutenance et d'y avoir assisté.

Contents

1	Introduction	1
1.1	Fault Attacks on Cryptographic Devices	3
1.1.1	Fault Definition	5
1.1.2	Secure and Non-Secure Systems	5
1.1.3	Different Types of Faults	6
1.1.4	Fault Analysis Methods	8
1.1.5	Fault Injection Techniques	10
1.1.6	Different Fault Models	15
1.2	Fault Attacks on the Advanced Encryption Standard	17
1.2.1	The Advanced Encryption Standard (AES)	17
1.2.2	Different Methods of Fault Attacks on the AES	19
1.3	The Physics of Fault Injection with a Laser	19
1.3.1	Laser Theory and Operation	19
1.3.2	Photoelectric Effect of Laser on Silicon	19
1.3.3	Different Parameters in a Fault Attack by Laser	21
2	Security Characterization	25
2.1	MicroPackS Security Characterization Laboratory	25
2.1.1	Laser Bench Characteristics	26
2.2	Circuit and Sample Preparation	27
2.2.1	Circuit Characteristics	27
2.2.2	Circuit Decapsulation	31
2.3	Security Characterization: First Mapping of Fault Injection Suscep- tibility	34
2.3.1	Laser Spot Parameters and Fault Injection	35
2.3.2	Exploration of our Circuit	37
2.3.3	Conclusions	39
3	Practical DFA by Laser on the AES	41
3.1	Problematic in Practical DFA by Laser	42
3.2	Piret-Quisquater's Differential Fault Analysis	43
3.2.1	Attack Scheme	43
3.2.2	Practical Experiment	44
3.2.3	Practical Experiment Without Memory Access	48
3.2.4	Conclusion	49
3.3	Giraud's Single-Bit Differential Fault Analysis	51
3.3.1	Attack Scheme	51
3.3.2	Practical Experiment	53
3.3.3	Conclusion	55
3.4	An Extended Single-Bit DFA for Multiple-Byte Faults	55

3.4.1	1 st Scheme for the Extended Attack	56
3.4.2	2 nd Scheme for the Extended Attack	57
3.4.3	Practical Experiment	58
3.4.4	Conclusion	59
3.5	Feasibility Comparison of Previous Attacks	60
3.5.1	Review of Piret-Quisquater's Experiment	60
3.5.2	Review of Giraud's Bit Experiment	64
3.5.3	Conclusion	65
3.6	An Extended Multiple-Byte DFA	68
3.6.1	Attack Scheme	68
3.6.2	Practical Experiment	71
3.6.3	A More Sophisticated Practical Experiment	72
3.6.4	Conclusion	74
3.7	Conclusion	74
4	Round Modification Attacks	77
4.1	Round Reduction Attacks	77
4.1.1	Previous Studies	77
4.2	Feasible Attack Models on our AES	79
4.2.1	Single Attack Scenarios	80
4.2.2	Cryptanalysis of the Main Attacks	87
4.2.3	Secondary Attack Scenarios	110
4.2.4	Cryptanalysis of Secondary Attacks	117
4.3	Conclusions	117
5	Countermeasures	119
5.1	Introduction	119
5.2	Countermeasures against Differential Fault Analysis	120
5.2.1	A Countermeasure against Fault Attacks on the Round Keys	120
5.2.2	A Countermeasure against Attacks on the KeyExpansion . . .	122
5.2.3	A Combined Countermeasure against DFA Attacks	123
5.2.4	Improvement of the Combined Countermeasure against DFA Attacks	124
5.3	Countermeasures Against Round Modification Attacks	127
5.3.1	An Unrolled AES Countermeasure	127
5.3.2	A More Secure Unrolled AES Countermeasure	128
5.4	A Combined Countermeasure against DFA and RMA	130
5.5	Conclusions	130
6	Other Security Perspectives	133
6.1	Introduction	133
6.2	A Very Close to Perfect Countermeasure against Power Analysis At- tacks	133
6.2.1	Introduction	133

6.2.2	Criteria for a Perfect Solution	134
6.2.3	A Very Close to Perfect Solution	135
6.2.4	Possible Attack on Our Countermeasure and Our New Solution	142
6.2.5	Conclusion	142
7	Conclusions and Perspectives	145
8	Résumé étendu en français (French Extended Abstract)	147
8.1	Introduction	148
8.1.1	Attaques en fautes des systèmes cryptographiques	149
8.1.2	Attaques en fautes sur AES	155
8.1.3	Physique de l'injection de fautes par laser	156
8.2	Caractérisation sécuritaire	157
8.2.1	Plateforme de caractérisation sécuritaire MircoPackS	157
8.2.2	Caractéristiques du banc laser	158
8.2.3	Préparation de circuit et des échantillons	159
8.2.4	Caractérisation sécuritaire : Première cartographie de la sus- ceptibilité d'injection de fautes	162
8.2.5	Exploration de notre circuit	163
8.3	Pratique de la DFA par laser sur AES	164
8.3.1	Problématiques de la pratique de la DFA par laser	164
8.3.2	DFA mono-octet de Piret et Quisquater	164
8.3.3	DFA mono-bit de Giraud	164
8.3.4	Résultats pratiques des DFA mono-octets et mono-bit	166
8.3.5	Une DFA multi-octets étendue	166
8.4	Attaques par modification de rondes	167
8.4.1	Modèles faisables d'attaque sur notre AES	168
8.4.2	Modèle d'attaques faisables sur notre AES	170
8.5	Contre-mesures	174
8.5.1	Contre-mesures contre l'analyse différentielle de fautes	174
8.5.2	Contre-mesures contre l'analyse de modification de rondes	174
8.5.3	Une contre-mesure combinée contre DFA et RMA	176
8.6	Autres perspectives de sécurité	178
8.6.1	Une contre-mesure efficace contre les attaques d'analyse de consommation	178
8.7	Conclusions et perspectives	180
A	Appendix RMA Results	183
A.1	Attacks on the Round Counter Value	183
A.2	Attacks on the Round Number Reference	189
	Bibliography	195

List of Figures

1.1	An overview of Differential Fault Analysis (DFA).	9
1.2	An overview of fault injection techniques.	11
1.3	Scheme of a FIB system.	15
1.4	AES general outline.	18
1.5	Architecture of a typical SRAM cell.	21
1.6	Absorption coefficient for silicon at various doping levels of p-type material.	22
2.1	MicroPackS laserbench.	26
2.2	Close-up on our circuit at the laserbench.	27
2.3	The structure of a command in T=0 protocol.	28
2.4	An overview of our AES implementation.	30
2.5	Chemical etching bench in MicroPackS laboratory and close-up on Nisene JetEtch II decapsulator.	31
2.6	A front side decapsulated sample of our chip by chemical etching.	32
2.7	Mechanical etching bench in MicroPackS laboratory equipped with an Ultra Tec ASAP-1 decapsulator.	33
2.8	A backside decapsulated sample of our chip by mechanical etching.	33
2.9	Main identified components on the original top metal overview photo.	34
2.10	Physical allocation of the SRAM's bytes on the front side of our decapsulated microcontroller, discovered by laser fault injection.	36
2.11	The vertical wire lines in the middle of our circuit.	38
3.1	1 μm & 10 μm laser spot diameters vs. technology scaling.	42
3.2	Propagation of a single-byte fault, injected at MixColumns input of 9 th round.	43
3.3	Effects of a single-byte fault at SubBytes input of 9 th round.	44
3.4	The exploration processes of our experiment are shown in three steps. First, we searched for K_8 bytes in SRAM. Then, we displaced the spot on SRAM in order to target only one K_8 byte and nothing on the following round keys. Finally, by adjusting precisely the timing of laser emission on t_8 , we logically injected only a single-byte fault on K_8 .	46
3.5	Piret-Quisquater's attack timing.	47
3.6	Effects of MixColumns on fault propagation for one faulty single-bit/byte round key on the temporary ciphertext at the end of each round and at the end of the algorithm.	48
3.7	Exploration process for Piret-Quisquater's DFA by blind experiment. The laser emission time was set to t_8 during exploration.	50
3.8	Giraud's bit DFA.	51

3.9	An intersection between three hypothesis sets for a K_{10} byte value.	52
3.10	Giraud's bit DFA.	53
3.11	Exploration process for Giraud's Single-Bit DFA. The laser emission time is set to t_9 during exploration.	54
3.12	Giraud's bit attack timing.	55
3.13	Exploration process with a big laser spot. The laser emission time is set to t_9 during exploration.	57
3.14	Exploration process and result classification. The laser emission time was set to t_9 during exploration. Then, the classification process excluded K_{10} -related faults.	58
3.15	Classification of faulty bytes on the ciphertext. Faults are separated into two classes of corresponding (K_9 -related) and non-corresponding (K_{10} -related) faults for Giraud's bit DFA.	59
3.16	A SRAM block diagram.	61
3.17	Four different sets of possible faulty bytes on K_8 . When more than one faulty byte is injected from an individual set into K_8 , their effects create multiple faults on the respective four bytes after MixColumns . Therefore, Piret-Quisquater's DFA process cannot find the key values.	61
3.18	Description of additional faulty bytes on K_8 : In part (a), when two faulty bytes are injected from the same set into K_8 , their effects change the content of the same column twice, after the next MixColumns . Therefore, Piret-Quisquater's DFA process cannot find the key values. Besides, part (b) shows that when two faulty bytes are injected from two different sets into K_8 , their effects speed up the Piret-Quisquater's DFA process.	62
3.19	Proper attack timing on the K_{10} for excluding the logical effect of eventual faults on the previous round keys.	68
3.20	Attack exploration for this extended multi-byte DFA. The laser emission time is set to t_9 during exploration.	71
3.21	The attack exploration for the more sophisticated case of our extended multi-byte DFA. The laser emission time is set to t_9 for the first encryption during exploration. The first and the second encryptions are shown in the top and the bottom parts, respectively.	73
4.1	Implementation of our AES algorithm.	79
4.2	Various moments of our AES algorithm execution.	80
5.1	Calculation of KxorRef bytes.	121
5.2	Key-protection countermeasure by Kxoring calculation and KeyTest operation in our AES implementation.	123
5.3	KeyCompare countermeasure.	124
5.4	Implementation of parallel encryption processes and KeyTest and KeyCompare operations in our AES implementation.	125

5.5	Implementation of the improved countermeasures in our AES implementation.	126
5.6	Implementation of the unrolled AES with <code>RoundTest1</code> and <code>RoundTest2</code> operations.	129
5.7	Combined countermeasures against DFA and RMA attacks in our AES implementation.	131
6.1	An Osram Orbeos OLED panel and a Sanyo Amorton AM-8801 photovoltaic cell.	136
6.2	Output and efficiency of the system by boosting the OLED panel over its authorized power limits.	138
6.3	Electronic schematic of the countermeasure.	139
6.4	The test bench.	140
6.5	The protected system and the attacker.	140
6.6	Close-up of the protected system.	141
6.7	Close-up of the thickness of the OLED panel and the photovoltaic cell (both with their protective glass).	141
6.8	Reset, power consumption and I/O signals observed on the card contacts at the reset moment.	142
6.9	Electronic schematic of the countermeasure and the card contacts, before and after applying the new solution.	143
8.1	Principe de l'analyse différentielle de fautes.	152
8.2	Aperçu des principales techniques d'injection de fautes.	154
8.3	Algorithme AES-128.	156
8.4	Plateforme laser MicroPackS.	158
8.5	Plateforme laser et circuit test.	159
8.6	Un aperçu de l'implémentation de notre AES.	161
8.7	Cartographie du microcontrôleur cible.	162
8.8	Mise en évidence de la taille de faisceaux laser d'1 μm et de 10 μm par rapport au progrès des technologies de fabrication.	165
8.9	Propagation d'une faute mono-octet à l'entrée du <code>MixColumns</code> de la ronde 9.	165
8.10	Attque DFA Giraud mono-bit.	166
8.11	Les instructions de notre algorithme AES.	169
8.12	Différents moments d'exécution de notre algorithme AES par rapport aux utilisation du compteur de rondes.	170
8.13	Processus parallèles de chiffrement et vérifications des clefs dans notre implémentation d'AES.	175
8.14	Implémentation de l'AES déroulé avec deux vérifications à chaque ronde.	176
8.15	Contre-mesure combinée pour des attaques DFA et RMA dans notre implémentation d'AES.	177
8.16	Système protégé par la contre-mesure.	179

8.17 Schéma électronique de la contre-mesure sans séparation des masses (haut) et avec séparation des masses (bas)	180
---	-----

List of Tables

1.1	Brief state-of-the-art of the most effective fault attacks on AES. . . .	20
2.1	Effects of laser energy on the number of injected faults into the SRAM using a spot size of $3.75 \mu m \times 3.75 \mu m$. The experiment is done on a front side decapsulated sample of our chip.	35
3.1	Potential faulty K_i s as a function of observed faulty ciphertext bytes. . . .	49
4.1	Effects of fault injection on the round counter according to the different stages of the AES algorithm execution	82
4.2	Comparison between a correct and a faulty execution.	83
4.3	Exploitable cases of the scenario I.	84
4.4	Effects of fault injection on the round number reference during different moments of AES middle or final round execution	86
4.5	The exploitable attacks of Scenario II.	86
4.6	The exploitable attacks of Scenario III.	112
5.1	An example for K_{RC} and $K_{xorRef_{RC}}$ values.	122
6.1	Some technical information about the Osram Orbeos CDW-031 OLED panel.	136
6.2	Maximum output power of photovoltaic cells when exposed to Osram Orbeos OLED panel light, boosted to $8.25W$	137
6.3	Results for different smart cards using the energy of the photovoltaic cell.	139
8.1	Conséquences de l'injection de faute sur le compteur de rondes lors des différents étapes d'exécution d'une ronde intermédiaire ou finale d'AES	171
8.2	Conséquences de l'injection de faute sur la référence du nombre de rondes lors des différents moments de l'exécution d'une ronde intermédiaire ou finale d'AES	172
8.3	Attaques exploitables sur CR ou R_{max}	173
A.1	All the single-bit attacks of scenario I.	184
A.2	All the single-bit attacks of scenario II.	190

Introduction

Contents

1.1	Fault Attacks on Cryptographic Devices	3
1.1.1	Fault Definition	5
1.1.2	Secure and Non-Secure Systems	5
1.1.3	Different Types of Faults	6
1.1.4	Fault Analysis Methods	8
1.1.5	Fault Injection Techniques	10
1.1.6	Different Fault Models	15
1.2	Fault Attacks on the Advanced Encryption Standard	17
1.2.1	The Advanced Encryption Standard (AES)	17
1.2.2	Different Methods of Fault Attacks on the AES	19
1.3	The Physics of Fault Injection with a Laser	19
1.3.1	Laser Theory and Operation	19
1.3.2	Photoelectric Effect of Laser on Silicon	19
1.3.3	Different Parameters in a Fault Attack by Laser	21

Cryptography is the study and the practice of methods for secret writing of messages. Its aim is hiding their meaning to everybody except an intended recipient who will be the only one who can uncover the secret and read the message [Mollin 2007] [Paar 2010].

Cryptography may be used to provide any of following properties:

1. Confidentiality: To prevent the unauthorized disclosure of data, only an authorized receiver should be able to extract the message contents from its encrypted form.
2. Integrity: The receiver should be able to determine whether he receives the original message or an altered version.
3. Authenticity: The receiver should be able to check from the message, the sender's identity and the message origin or the path it travelled.
4. Non-repudiation: The sender should not be able to deny sending the message.

Modern cryptography is based on mathematics, computer science, and electrical engineering. It includes symmetric and asymmetric methods. In the first group, messages are encrypted and decrypted using a unique secret key that provides security for the sender and the receiver. Besides, in the asymmetric methods, encryption and decryption are done using two separate keys, one used by the sender to encrypt the message, the other one used by the receiver to decrypt the cipher.

A *cryptographic system*, according to [Denning 1983] and [Van Tilborg 2005], is a message ciphering and deciphering system composed of an *encryption algorithm*, a *decryption algorithm* and a well-defined triple of text spaces:

1. $\mathcal{M}$: plaintexts space.
2. $\mathcal{C}$: ciphertexts space.
3. $\mathcal{K}$: keytext space.

The encryption algorithm $\mathcal{E}$ transforms any plaintext by using the given key to a corresponding ciphertext:

$$\mathcal{E}_{K_E} : \mathcal{M} \rightarrow \mathcal{C} \quad \text{where} \quad K_E \in \mathcal{K}$$

The decryption algorithm $\mathcal{D}$ also maps the ciphertext to its corresponding plaintext using the proper key:

$$\mathcal{D}_{K_D} : \mathcal{C} \rightarrow \mathcal{M} \quad \text{where} \quad K_D \in \mathcal{K}$$

In asymmetric cryptography, the key used for decryption is different from the encryption key. Moreover, symmetric cryptography employs a unique key for both transformations: $K_E = K_D$.

The encryption and the decryption transformations are usually uniquely determined as an *injective function*.

The cryptographic systems are also called *cipher systems* or more briefly *cryptosystems*. They were originally performed by hand methods using pen-and-pencil calculations. In the 18th century, mechanical machine methods were introduced to speed up cryptography. About one century later, they were replaced by the generation of electromechanical devices. The German Enigma and the American ECM Mark II were two famous models of these crypto-machines used during the Second World War. Nowadays, encryption and decryption have evolved and emerged from the mechanical era and they are mostly performed by software solutions or hardware accelerators.[Konheim 2007]

[Denning 1983] lists three general requirements for any cryptosystem:

1. Efficiency of encryption and decryption transformations for all the keys.
2. Ease of use of the system.

3. Dependence of the cryptosystem security on only the secrecy of the key and not on the secrecy of the encryption and decryption algorithms.

He also describes two secrecy requirements for cryptosystems:

1. For any intercepted ciphertext C , even if its corresponding plaintext M is known, systematic determination of the decryption key K_D must be computationally infeasible by cryptanalysis.
2. For any intercepted ciphertext C , systematic determination of the corresponding plaintext M must be infeasible by cryptanalytic computations.

The confidentiality requirements are just for protecting the decryption transformations. If the encryption transformation does not give away secrets about the decryption, it can be public knowledge. In symmetric methods, the same encryption key is used for decryption. Therefore, the encryption key must be protected. In asymmetric cryptography, the encryption key can be exposed.

So, the security of the cipher is based on the security of the decryption key in modern cryptography. According to one of Auguste Kerckhoffs' principles, presented in 1883, a cryptosystem should be secure even if everything about the system, except the key, is public knowledge [Kahn 1973].

Consequently, today's cryptosystems contain secret keys for cryptographic algorithms used to protect confidential information or to provide authentication mechanisms. For this reason, they are always the subject of much research aimed at improving their security and resistance to any unauthorized interference.

1.1 Fault Attacks on Cryptographic Devices

Attacks on cryptographic circuits can be categorized into two main families:

1. Cryptanalytic or Mathematical Attacks

These attacks search for vulnerabilities in a cryptographic schema or algorithm in order to deduct the keys by mathematical methods. When an opponent is not able to find any weakness in a cryptosystem that could help him perform a cryptanalytic attack, he may use an *exhaustive key search* [Paar 2010].

An exhaustive search or a *brute-force attack* for finding the key is a cryptanalytic attack that can, in theory, be used for finding a key that maps a plaintext to its corresponding ciphertext. It requires checking all possible keys until the correct one is found. In practice, it requires checking, on average, half of the entire key search space.

The key length of reliable cryptographic algorithms increases continually beyond the computer's capability of calculating and finding the keys. So, a

brute-force search for the keys cannot give any answer in a reasonable amount of time, except if it has been applied as a complement of another attack that can reveal a significant part of a key.

For instance, the *Data Encryption Standard*, called DES, was approved by former US National Bureau of Standards as an official *Federal Information Processing Standard* (FIPS) for the United States in 1976. DES is a block cipher that encrypts messages with a 56-bit key. It was considered as a secure encryption method at that time. But, as a result of the growth of computation capabilities, the US National Institute of Standards and Technology (NIST) announced in 1997 that they wished to choose a successor to DES. They mentioned that the new standard will be known as *Advanced Encryption Standard* or AES. They also remarked that the AES will be “an unclassified, publicly disclosed encryption algorithm capable of protecting sensitive government information well into the next century”. Therefore, the new encryption algorithm is chosen with 3 key sizes of 128, 192 and 256-bit after more than three years of studies and discussions.

The growth of minimal key size of the US encryption standard from 56 bits to 128 bits over the course of 25 years shows the importance of key size for information security. This example is much more obvious when we consider that each additional bit doubles the required computations in an exhaustive search.

Another significant example is the success of new research reported in [Bogdanov 2011]. The method decreases the complexity of exhaustive search for AES key variants for only two bits. In the reported research, the complexity is reduced from 2^{128} , 2^{192} and 2^{256} to $2^{126.1}$, $2^{189.7}$ and $2^{254.4}$ respectively. However, the exhaustive search with this remaining complexity still needs thousands of years given the current computation capabilities.

2. Hardware Attacks

This large family of attacks targets the hardware, *i.e.* the physical implementation of crypto-algorithms on integrated circuits. It includes two main categories:

(a) Side-Channel Analysis

These attacks are based on the analysis of any information leakage from a circuit during the encryption operations. This information leakage may be related to sensitive data processing and may reveal the secret key. For instance, the running time of a cryptographic circuit may leak information about the secret parameters involved, in the calculation process, if this time is data dependent [Van Tilborg 2005]. This idea was first introduced by Paul Kocher in [Kocher 1996].

Moreover, the electrical consumption of a microcontroller can be measured with a resistor inserted between its V_{CC} or its ground pin and the actual V_{CC} or ground. So, the supplied current will be transformed into a voltage easily monitored with an oscilloscope [Van Tilborg 2005]. This other kind of Side-Channel Analysis was also introduced by Paul Kocher later in [Kocher 1999].

The measurement of other parameters, such as electromagnetic radiation, heat emission or even photon emission of a circuit allows the attacker to conduct Side-Channel Analysis [Gandolfi 2001] [Joye 2005].

(b) Fault Attacks

They consist in using hardware malfunction to infer secrets from the target's faulty behavior or outputs. [Bellcore 1996] and [Boneh 1997] had reported in 1997 the possibility of secret leakage by physical perturbations. [Biham 1997] has presented a differential analysis method to exploit such faults. These attacks can be performed in different physical manners as reported in [Tria 2000]. Nowadays, different analysis methods are developed to reveal secrets from faulty behavior or outputs.

This thesis work is dedicated to the study of fault attacks: How they are injected and how faulty outputs are used to endanger the secrecy of cryptographic devices. More insights into Side-channel analysis methods could be found by the reader in the given references.

1.1.1 Fault Definition

A fault in a cryptographic system refers to an accidental or an intentional condition that causes the encryption or decryption process to deviate from its correct execution or result. In this case, the cryptographic system may act abnormally or the result of encryption or decryption may be incorrect, thus considered as *faulty*.

A faulty execution or result is considered *reproducible* if it occurs consistently under the same circumstances.

1.1.2 Secure and Non-Secure Systems

Conventional integrated circuits have functioning limits according to their physical conditions. For instance, a full-adder as part of a microcontroller may take more time to perform an intermediary addition at extreme temperature conditions. Consequently, the final result may be released before end of this intermediary operation. In this case, the final result will be faulty.

On the other hand, these conventional integrated circuits have a huge leakage of sensitive information that permits side-channel analysis.

These kinds of integrated circuits are considered to be *non-secure* for cryptographic systems and they are strictly not recommended for these purposes.

Furthermore, there are specific integrated circuits for cryptographic systems that resist better at physical extreme conditions. When there is a risk of faulty functioning, behavior or result, they interrupt their operations or release an intentional and pre-programmed fully faulty result that does not permit the deduction of any sensitive information. These circuits are called *secure systems*. This kind of protection may be applied to side-channel leakages.

For example, a bank card with an integrated chip is a kind of secure system. According to ISO/IEC 7816-3 standard, the chip operations interrupt when the V_{CC} increases or decreases more than 10% from its original value of 5.0 V [ISO/IEC 7816-3 2003].

Besides, commercial bank cards have data masking protection which functions by random calculations on their chips. So, predicting the manipulated data on the power consumption curve becomes very difficult or even infeasible.

This kind of protection against side-channel analysis or fault attacks is named *countermeasure*. It refers to an action or technique dedicated to preventing an attack from succeeding. A countermeasure may act by eliminating or preventing the threat, by neutralizing the harm it can cause and, if possible, reporting it. So the proper action must be taken in order to protect the sensitive information.

According to [Standaert 2009], no single technique can provide perfect security. Protecting implementations against physical attacks consequently seeks to make the opponent's task harder. The implementation cost of a countermeasure is also important and must be evaluated in comparison to the additional security obtained.

In this thesis, our experiments are done in the context of characterization of a non-secure circuit. At the end, in chapter 5, we will present some countermeasures.

1.1.3 Different Types of Faults

Faults on the electronic circuits can be classified into three categories, according to their persistence:

- **Provisional or transient faults:** These faults are temporary or short-term. As the fault injection interrupts, the provisional faults disappear. So, after some time has elapsed, the chip recovers its normal execution without circuit reset. For instance, heating a circuit creates faults by lengthening propagation times; the circuit resumes its correct functioning after temperature decrease.
- **Permanent faults:** The permanent faults are persistent but reversible. As the corrupted area is modified or changed by another part of the circuit or as the circuit is reset, these faults disappear. Thus, they are not destructive and don't damage the circuit. For instance, a fault injected on a SRAM cell persists until memory rewrite or circuit reset. Besides, a fault injected on a program code stored in a non-volatile memory has a more persistent effect. In this case, a circuit reset is not sufficient to recover the memory content. Nonetheless, the memory must be rewritten.

- Destructive faults: The interferences may create a perpetual defect on hardware. Once inflicted, such destructions affect the chip's behavior permanently. For example, a laser emission with a high energy level on a memory cell may permanently destroy some memory cells. In this case, the memory cells can no longer be rewritten or recovered by circuit reset.

Different physical reasons are identified as the origin of faults in each class [Bar-El 2006] [Barenghi 2012]. We summarize them in the subsequent section. Now, we are going to describe briefly the main permanent and destructive faults in secure circuits.

1.1.3.1 Permanent Faults

Effects of permanent faults are reversible. After a system reset or when the fault's stimulus is interrupted, the circuit will recover its original behavior. Two different kinds of provisional faults are considered in this class [Bar-El 2006]:

- Single-event upsets (SEUs): This kind of fault was first noticed as an effect of cosmic rays during a space mission in 1975. Research then began on injection mechanisms of such faults into the circuits. SEUs consist in a cell's logical state flipping to a complementary state without any damage to the circuit. If the fault is produced in a system that recovers its original values after a reset, its effect is temporarily. [Richter 1987] has reported in 1987 that SEUs can be created using focused laser beams. So, much research has been focused on the use of laser beams for injecting faults [Darracq 2002].
- Multiple-event upsets (MEUs): They consist of several SEUs occurring simultaneously. So, MEUs can be considered as a generalization of SEUs. By growing the integration density, the risk of generation of such faults can be increased.

1.1.3.2 Destructive Faults

Destructive faults are due to an effect on the circuit that remains permanent and creates lifetime faulty value or behavior. Different types of faults are included in this class [Bar-El 2006]:

- Single-event snap back faults (SESBs): These kinds of faults are created by the self-sustained current by the parasitic bipolar transistor in channel N of MOS transistors. It seems that they do not occur in low supply voltage devices.
- Single-event latch-up faults (SELS): A latch-up consists in the activation of a parasitic thyristor structure formed in CMOS circuits. The transient current induced by a laser beam, for example, may activate the parasitic thyristor resulting in a high current flow.

After these brief descriptions of different categories of faults, according to their persistence, we review different types of fault analysis methods in the next section.

1.1.4 Fault Analysis Methods

Different analysis methods for fault attacks have been figured out by the researchers. These analysis methods usually require injection of transient or permanent faults. Otherwise, if the opponent creates destructive faults, the procedure cannot be repeated on the circuit.

- **Round Reduction (RR):** Many cryptographic algorithms are based on repetition of identical sequences of transformations, called *rounds*. A significant part of these algorithms' strength against cryptanalysis is based on their repeated rounds. Any decrease in the number of rounds reduces their security.

The Round Reduction belongs to the family of attacks by algorithm modification. For instance, suppose an attack that makes a jump to the end of the algorithm after the execution of just a few instructions or after just the first round. Thus, the remaining encryption processes are skipped and the final ciphertext is the product of a few algorithmic processes that may easily reveal the key.

Principle of Round Reduction is based on decreasing the number of rounds in an algorithm in order to facilitate subsequent cryptanalysis. This method was first presented in [Choukri 2005]. It illustrates that a transient glitch on the V_{cc} may change the round counter value of a repetitive cipher. The opponent may break the algorithm execution at end of the first round. In this case, the cryptanalysis will be very fast and easy. Its complexity no longer corresponds to the cryptanalysis of correct execution for the entire 10 rounds of the reported algorithm.

[Choukri 2005] reported this attack on a PIC16F877 microcontroller. Other works are reported in [Monnet 2006] and [Park 2011]. We will review RR attacks in chapter 4.

- **Differential Fault Analysis (DFA):** This method is based on gaining some insight into the secret data handled by the circuit and then finding the secret key by comparing faulty ciphertexts with the corresponding (correct) ciphertexts. An overview of this method is shown in the figure 1.1.

The first alert of the possibility of using faults for breaking cryptosystems is reported by Bellcore (Bell Communications Research, Inc.) in [Bellcore 1996] and then in [Boneh 1997]. The first structured analysis method is presented as DFA in [Biham 1997]. Since the publication of [Biham 1997], many DFA methods were developed on different cryptographic algorithms.

We will describe two main DFA methods on AES in chapter 3.

- **Safe-Error Analysis (SEA):** As opposed to Differential Fault Analysis which searches for faulty ciphertexts, this analysis method searches for the existence

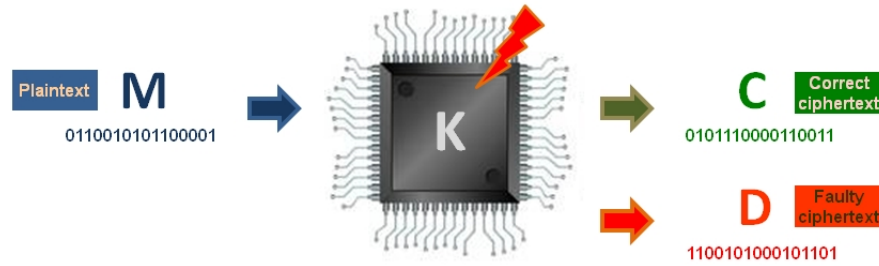


Figure 1.1: An overview of Differential Fault Analysis (DFA).

of any behavioral difference. A fault attack, may release an alarm or stop the operations. These signs of behavioral differences as compared to normal execution may lead to finding secrets in the circuits. [Blömer 2003]

The first SEA is presented in [Yen 2000]. It consists in the injection of a fault on a temporary register value and then observing the consequences on the output. [Yen 2000] illustrated an analysis method with simpler cryptanalytic complexities in comparison with some other techniques.

One year later, [Yen 2002] reported a safe-error based attack by inducing a temporary random computational fault in addition to a temporary memory fault, which was reported in [Yen 2000]. The advantage of the new attack is again simple cryptanalytic complexity. Some publications, such as [Lu 2005], distinguish between the two attacks, by considering the first method as a *Memory* or *M Safe-Error* that targets memory or registers content and the second one as a *Computational* or *C Safe-Error* Analysis focusing on the operations. Moreover, [Kim 2007] reports vulnerability of a smart card implemented with a hardware modular multiplier to another type of SEA.

In 2003, Blömer and Seifert presented two SEA on an AES algorithm in [Blömer 2003]. The first attack determines entirely the 128-bit secret key of a sealed tamper-proof smartcard by generating 128 faulty cipher texts. The second attack observes any differences in execution time of a particular operation in presence of faults and reveals again the secret key by using 256 encryptions.

- **Differential Behavioral Analysis (DBA):** This method is based on correlating a functional model parametrized by the value of a partial key to behaviors of the device in presence of faults. This analysis method, presented in [Robisson 2007], combines Safe-Error Analysis and Differential Power Analysis.

SEA methods are still applied to very specific vulnerabilities. By mixing the principles of SEA and the probabilistic treatment of DPA, [Robisson 2007] reported a more realistic approach.

- **Fault Sensitivity Analysis (FSA):** This new method reported in [Li 2010] is based on the analysis of critical conditions when a faulty output begins to exhibit some detectable characteristics.

According to [Li 2010], these characteristics are related to the sensitive data and can be used to retrieve the secret key. For instance, an increase in the clock frequency represents a critical condition value when faulty operations begin to occur. So, this method analyzes the sensitive conditions of fault occurrence instead of DFA methods that compare values of corresponding faulty and correct ciphertexts.

1.1.5 Fault Injection Techniques

Different fault injection techniques have been developed in order to intentionally alter the correct functioning of a computing device. However, the use of these methods depends on many parameters, especially the degree of technical skill and financial possibilities of the opponent as well as his proficiency in the implementation and the characterization of achievable faults [Bar-El 2006][Barenghi 2010]. Figure 8.2 shows an overview of the most well-known techniques for fault injection:

- Voltage glitch and clock modification may be applied to the corresponding pins on the chip.
- Temperature increase and electromagnetic impulsion might be applied without chip decapsulation.
- High-energy light emission and laser radiation cannot alter the chip functioning when it is not decapsulated. Electromagnetic impulsions may be applied with more accuracy on the decapsulated chips.

As the range of fault injection techniques is wide and is getting wider, we classify them according to their cost. We also indicate the degree of required knowledge and technical skill for the implementation by the opponent.

In fact, the opponents may be considered as two significant types:

1. Curious individual people with a low or medium knowledge of a system and a modest personal budget.
2. Expert people with a medium or good knowledge of a system and a high personal budget. This category may also include a group of experts or companies with a good budget that help each other to acquire a very good knowledge of a system.

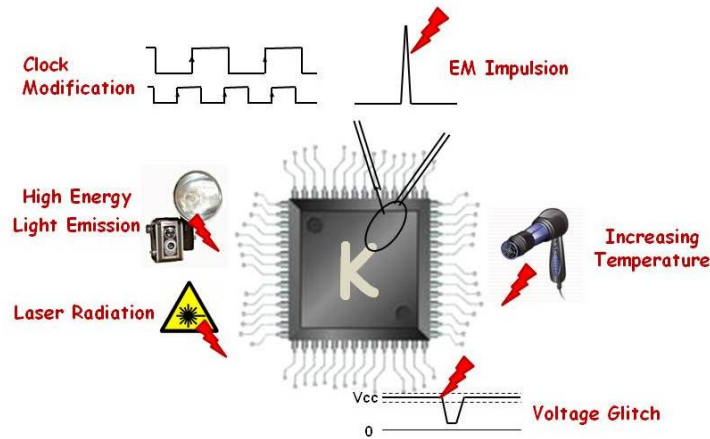


Figure 1.2: An overview of fault injection techniques.

1.1.5.1 Low cost fault injection methods

[Barenghi 2012] considers as *low cost* any fault injection technique that needs to less than 3000 USD of equipment in order to set up the attack. This budget is estimated for the ability of a single motivated opponent. So, these fault injection methods are usually intended as a serious threat. Consequently, the implementation of recent secure chips has mostly proper countermeasures against these threats which render them unable to alter the security.

1. Underpowering the circuit during execution may cause a processor to interrupt or skip instruction. By running the chip with a decreased power supply, the attacker is able to insert transient faults starting from single bit errors and becoming more numerous as the supply voltage gets lower.

This method is reported to be effective on large integrated circuits. As reported in [Barenghi 2009], underpowering of the circuit entered delays in the set-up time of an ARM9 processor's logical gates. Therefore, faults are injected into the circuit computations. Another successful attack is reported in [Selmane 2008] on a symmetrical encryption algorithm, embedded in a smart card.

This technique, performed by using an accurate power supply unit requires the opponent to be able to penetrate into the power supply line of the device and connect it to his own power supply unit. So, underpowering the circuit requires only basic skills and can be easily achieved without evidence of tampering. Detailed knowledge of the device's implementation is not required.[Barenghi 2012]

2. Tampering the clock signal is another option for causing faults. In this attack, the clock period is made shorter than required by the timing constraints for

normal functioning. So, the expected operation of this manipulated single signal cannot be executed correctly.

If, for instance, a data read is requested from the memory and it does not have the time to send the content onto the bus, a data misread happens. Or, if before finishing the execution of a current instruction, the shorter clock signal causes jumping to the next instruction, the processor misses the effect of the current one.

According to [Amiel 2006], premature rising of a single clock signal may cause multiple errors corrupting a stored single byte or multiple bytes. [Agoyan 2010b] reports how, by shortening a single clock pulse, an accurate and reproducible single-bit fault can be injected during encryption on an FPGA platform.

These errors are transient and can be induced without any tamper evidence. Nevertheless, the opponent needs to have direct control of the clock line. For the chips that generate their own clock signal internally, disconnecting the clock line and performing such an attack seems infeasible [Barengi 2012].

3. Altering the environmental conditions is another possibility for injecting faults. For instance, by increasing the temperature, it is possible to create several multi-bit errors in DRAM memories. [Govindavajhala 2003] illustrates a thermal fault injection attack against DRAM chips of a common desktop computer. Increasing the working temperature to 100°C caused approximately 10 flipped bits per each 32-bit word. The number of faulty words was also about 10. In this test, a 50 watt light bulb and thermometer are used as fault injection equipment.

Heating a circuit lengthens propagation delays in CMOS logical circuits. If the propagation delays in a circuit become greater than the clock period then faults appear. [Dutertre 2010] reported successful fault injection into an embedded cryptographic algorithm on an FPGA. The circuit was heated to 210°C using an electrical heater with compressed air.

For this kind of attack, a minimal technical skill suffices. However, the possibility of damaging the circuit by excessive temperature can still be considered as a potential risk.

4. Creating strong electromagnetic (EM) disturbances near the circuit is also another practical way to induce faults. [Schmidt 2007] illustrates an effective EM attack against an 8-bit microcontroller. This attack is applied by using a spark generator, as an EM disturbance source, located very close to the targeted chip. [Schmidt 2007] also reports a more effective attack when the targeted chip is decapsulated. In these attacks, a simple piezoelectric gas lighter is used as a spark generator and it is held directly above the device.

Besides, in this kind of attack, it is necessary to protect the components which should not be affected by the EM fault injection. A properly grounded metal plate or mesh can be used as a suitable shield. In [Schmidt 2007], all the parts of the microcontroller which did not need to be disturbed were properly shielded through grounded aluminium plates.

5. Using high energy light sources on a decapsulated chip is another possible technique. An UV lamp or camera flash can be used to inject faults. According to [Schmidt 2009] an effective attack is accomplished by using UV irradiation on four different depackaged microcontrollers. The irradiation on the silicon surface, can cause the blanking of erasable EPROM and flash memory cells where stored constants are needed for a cryptographic algorithm execution. [Schmidt 2009] reports that a progressive erasing of all the non-volatile memory cells, as well as resetting the internal protection fuses, can be done, as long as the irradiation process continues.

1.1.5.2 High cost fault injection methods

According to [Barenghi 2012], a second class of fault injection techniques exists for opponents with a bigger budget. These techniques are developed in response to the need to have direct access to the silicon die and the ability to accurately target individual circuits. They require a budget above the aforementioned 3000 USD, sometimes going up to millions. These methods are always combined with invasive methods that leave evident traces of tampering. However, they are usually powerful and offer a good feasibility for successful attacks.

1. A powerful and accurate focused light spot is a simple method to induce alterations in the behavior of one or more logic gates of a circuit. For obtaining an accurately focused light beam from a camera flash, the use of a precision microscope is required. Otherwise, the use of low-quality lenses results in the diffraction of the light beam [Barenghi 2012].

[Skorobogatov 2003] and [Skorobogatov 2005] report successful targeting of the SRAM cells using this method. The attack caused bit-flip on the targeted SRAM cell of a microcontroller.

In [Giraud 2005], by using a microscope, a modified camera flash and a computer, an effective attack against a cryptographic algorithm on an 8-bit smart card is reported. The attack is done on a decapsulated microcontroller running an embedded AES algorithm.

Furthermore, the width of the gate dielectric in current fabrication technologies is more than 10 times smaller than the shortest wavelength of visible light. So, theoretically, it is no longer possible to hit a single SRAM cell on an up-to-date circuit with current etching technology.

2. Laser emission on a decapsulated chip is another way to generate faults. Using a laser beam allows an opponent to target more precisely a small circuit area. It can be considered to be the most straightforward refinement of the previous technique [Barengi 2012]. The fault model is also similar to the previous one with more feasibility of creating faults and reproducibility.

In the laser emission technique some parameters such as the wavelength of the laser beam and the exposed side of the chip are important. We will review them in section 1.3.

Currently, commercial fault injection workstations are available that are composed of a laser emitter, focusing lens and placement surface with stepper motors to achieve an accurate targeting of the beam. They exist in different size, from portable models to big laser benches with widely different accuracy and also with prices ranging from a few thousand to several hundred thousand USD.

This method also has some limitations: The incident area on a targeted chip cannot be reduced to a smaller size than the laser wavelength. Therefore, depending on the fabrication technology, the incident area is physically larger than one gate in current and oncoming technologies. We will illustrate this point in 1.3.

3. X-ray emission may also induce faults on electronic circuits and alter their functioning. Some research, such as [Schwank 2006] has confirmed the impact of X-rays, coming from cosmic rays, on aircraft electronic devices. In addition, [Bar-El 2006] reported it as a fault injection technique. However, it seems that there is not any research about its intentional impact on the secure systems.

According to [Otto 2004] and [Govindavajhala 2003], the energy level in standard commercial X-ray devices, such as an airport baggage scanner, is much lower than the required threshold to inject faults on small electronic devices. But, a “hard” X-ray source with a high-level energy may induce voluntary faults on a small data object, such as a DRAM circuitry [Govindavajhala 2003].

To examine the feasibility of this technique, the opponent needs some equipment that is more expensive than what is used in low cost methods. However, as the X-rays have a very short wavelength, in the range of 0.01 to 10nm, it might be possible that they could inject accurate faults. Future research will determine this.

4. Using Focused Ion Beams (FIB) permits an opponent to perform the most accurate and powerful fault injection attack at the present time [Barengi 2012]. But, this technique is expensive and FIB workstations cost several thousand to several million USD. FIBs permit the opponent to arbitrarily modify a circuit’s structure, reconstruct missing buses, cut existing wires, mill through layers and rebuild them. Such FIB workstations are commonly used to debug

and patch chip prototypes, or to reverse unknown engineering designs through adding probing wires to otherwise inaccessible parts of the circuit.

FIBs may also be used for probing logic levels of an integrated circuit interconnections. This is not fault injection, but rather *circuit spying*.

[Melngailis 1986] proposed in 1986 to explore the ion milling capability of FIBs as a tool for integrated circuit reconstructing. This paper illustrated two methods for cutting and joining conductors on the circuits. Afterwards, other research, such as [Anderson 1996] and [Kömmerring 1999], confirmed the possibility of performing attacks by FIBs on secure devices. [Torrance 2009] reports a successful reconstruction of an entire read bus of a memory containing a cryptographic key without damaging the content of the memory.

Figure 1.3 shows the scheme of a FIB system, taken from [Fibics]. The most advanced FIB systems can operate at a precision of $2.5nm$. This accuracy is less than a tenth of the gate width of the smallest etchable transistor in current fabrication technologies. Besides, FIB workstations and their consumables are very costly. They also require a strong technical background to fully exploit their capabilities. According to [Barenghi 2012], the only limit to the FIB technology is the diameter of the atoms whose ions are used as a scalpel. Currently, the most common choice is Gallium, which sets the lower bound to roughly $0.135nm$.

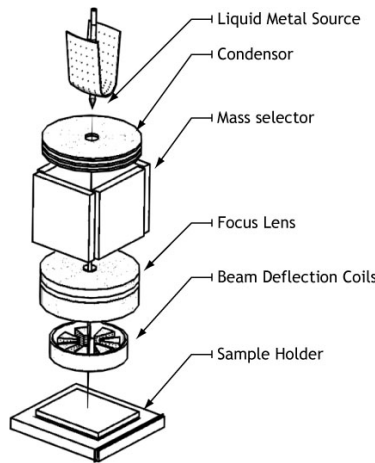


Figure 1.3: Scheme of a FIB system.

1.1.6 Different Fault Models

The injected faults on the circuits may be described with different fault models. For describing these differences, we consider $T_1 = \{b_1, b_2, \dots, b_n\}$ as the initial values of

an arbitrary set of targeted bits. Let $T_2 = \{b'_1, b'_2, \dots, b'_n\}$ be values of T_1 after a fault attack. Now, we review the effect of different fault models on the targeted set:

- **Bit-flip or Bit inversion:** When the values of targeted bits are changed to their opposite values, we consider the fault type as *bit-flip* or *bit inversion*, if and only if:

$$\forall i : 0 \leq i \leq n \quad ; \quad b'_i = 1 - b_i \quad (1.1)$$

- **Stuck-At:** In this fault model, the targeted bits are set permanently to their previous value. Therefore, even if new values must be affected to the targeted bits, the memory write operation cannot change them. This effect is usually considered as a destructive fault due to a wire, gate or memory cell damage, but it might be a permanent fault that disappears after a circuit reset.

The fault model is considered as *stuck-at 0*, if and only if:

$$\forall i : 0 \leq i \leq n \quad ; \quad b_i = b'_i = 0 \quad (1.2)$$

Otherwise, the fault model is *stuck-at 1*, if and only if:

$$\forall i : 0 \leq i \leq n \quad ; \quad b_i = b'_i = 1 \quad (1.3)$$

In this category of faults, the values of targeted bits are usually unknown to the opponent before and after the attack. A stuck-at fault has a noticeable effect only when it must be rewritten to its opposite value. At this point, it may create a change in the system behavior or results [Otto 2004].

- **Random:** When the value of at least one of targeted bits is changed, but the value changes are random. In other words, the fault model is *random*, if and only if:

$$\begin{aligned} \forall i : 0 \leq i \leq n \quad ; \quad b'_i \in \{0, 1\} \\ \text{and} \\ \exists ! j : 0 \leq j \leq n \quad ; \quad b'_j \neq b_j \end{aligned} \quad (1.4)$$

- **Set or Reset:** In this fault model, the targeted bits are set or reset to whatever is their previous value. The fault model is considered as *set*, if and only if:

$$\forall i : 0 \leq i \leq n \quad ; \quad b'_i = 1 \quad (1.5)$$

Otherwise, the fault model is *reset*, if and only if:

$$\forall i : 0 \leq i \leq n \quad ; \quad b'_i = 0 \quad (1.6)$$

Between these fault models, the random faults are usually considered to be the most realistic.

1.2 Fault Attacks on the Advanced Encryption Standard

1.2.1 The Advanced Encryption Standard (AES)

AES is a symmetric method and is based on Rijndael cipher [NIST 2001]. AES performs message encryption by processing data blocks of 128 bits at input and output. According to the size of the key, AES performs the encryption (or the decryption) in 10, 12 or 14 rounds (in addition to a short initial round) using a key of 128, 192 or 256 bits, respectively. The algorithm includes two separated processes: One for the **KeyExpansion** (or the **KeyScheduling**) to derive the round keys from the secret key and another one for the **DataEncryption**.

Decryption is also divided into two separate processes: One for the **KeyExpansion** and another one for the **DataDecryption**.

AES can grant a high level security using a reasonable calculation time. So, AES was quickly adopted for many systems and products after NIST validation in 2001. Thus, many types of attacks have been studied by researchers with the intention of improving AES implementations by suitable countermeasures.

In June 2003, US National Security Agency (NSA) announced that “*The design and strength of all key lengths of the AES algorithm (i.e., 128, 192 and 256) are sufficient to protect classified information up to the SECRET level. TOP SECRET information will require use of either the 192 or 256 key lengths*” [CNSS 2003]. However, it noticed that “*The implementation of AES in products intended to protect national security systems and/or information must be reviewed and certified by NSA prior to their acquisition and use*” [CNSS 2003].

Therefore, detection and mitigation of any potential threat is vital for AES systems security. Today, a significant part of research in cryptography is focused on improving the defense of cryptographic algorithms and systems against any eventual attack.

For the initial round in AES, the algorithm uses the secret key as the round key; but for each following round, the corresponding round key is calculated from the previous one. Figure 1.4 shows the different operations of the AES-128 algorithm. Hereafter, we use AES to refer to AES-128 and we use the “*K*” prefix plus the number of a round to refer to a round key (e.g. “*K*₉” for the round key of the 9th round). For other AES versions, we will mention their key lengths (i.e. AES-192 and AES-256).

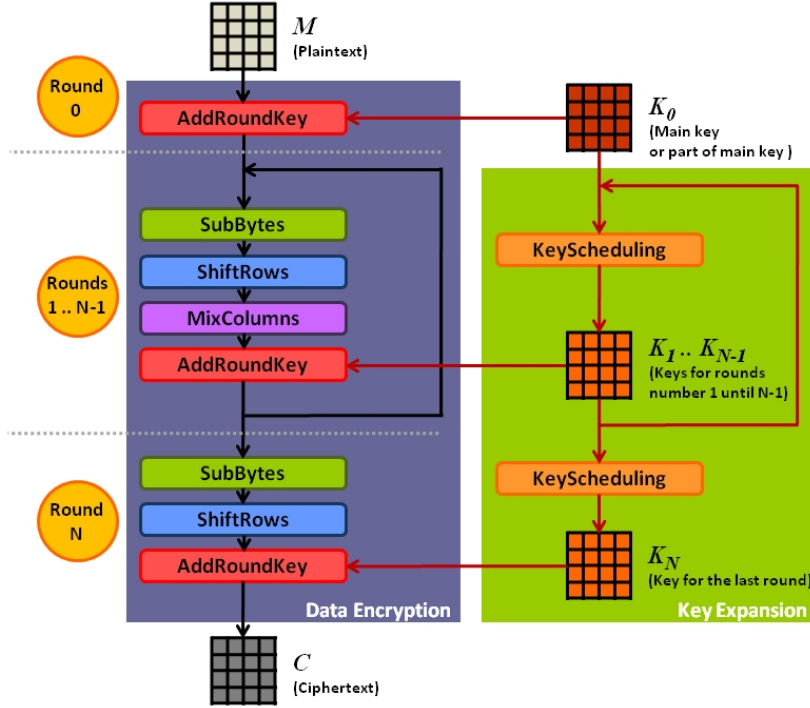


Figure 1.4: AES general outline.

To encrypt a plaintext, namely M , according to the implementation of AES, usually at the beginning of algorithm execution, all the round keys are computed from the main key and are stored in the memory. Then, the encryption process begins and takes separate blocks of 16 bytes (128 bits) from M as input and puts each block in a matrix of 4×4 bytes. Each round of the algorithm, except the initial round and the final round, includes 4 steps: At the beginning, it exchanges the value of each matrix element, i.e. one byte value, with the corresponding value in a substitution table (**SubBytes** or **SB**). Then, it executes a rotational operation on the matrix rows (**ShiftRows** or **SR**). In the third step, the algorithm applies a linear transformation to each element and combines it with other values in the same column with a different coefficient of 1, 2 or 3 for each element (**MixColumns** or **MC**) under the specific rules of $\text{GF}(2^8)$. This step guarantees the distribution of the information of each byte on 4 bytes and increases security of encrypted messages. In the last step of each round, a bitwise xor operation is performed between the value of each element and the corresponding byte on the round key (**AddRoundKey** or **ARK**).

Currently, AES encryption is widely used for governmental, military and commercial purposes. Therefore, it has opened a new and large domain of research on the security of cryptographic circuits.

1.2.2 Different Methods of Fault Attacks on the AES

Different types of fault attacks on AES have been studied by researchers. They can be categorized into DFA, SEA and RR groups. Depending on the theoretical model and the physical implementation, the target can be a temporary ciphertext, a round key, the `SubBytes` table or the round counter. In addition, the theoretical model defines the focalization (e.g. a selected bit or any reachable bit on a selected byte) and the fault model. Often the theoretical attack requires a specific kind of fault model, such as flipping a bit value, where the bit value is stuck at 0 or 1, or where a random faulty value is injected into a byte content. Table 1.1 shows a brief state-of-the-art of the most effective fault attacks on AES.

The feasibility of performing physical attacks depends on different parameters. A single-bit level attack on a chosen byte can be considered as a subset of any possible faulty values on a byte. In a single-byte attack, any of $2^8 - 1$ different faulty values may happen. But, a single-bit attack requires only one bit difference, and thus, a subset of 8 options.

Furthermore, the time constraints are at stake. The injection of a required fault is always needed in a restricted time of the algorithm execution. In chapter 3, we will review Piret-Quisquater and Giraud's bit attacks as two main DFA on AES and will discuss in more detail the attack time constraints.

1.3 The Physics of Fault Injection with a Laser

1.3.1 Laser Theory and Operation

Laser (Light Amplification by Stimulated Emission of Radiation) is a stimulated-emission electromagnetic radiation in the visible or the invisible domain. Laser light is monochromatic, unidirectional, coherent and artificial (i.e. laser does not spontaneously exist in nature). Laser light can be generated as a beam of very small diameter (a few μm). The beam can pass through various material obstacles before impacting a target during a very short duration.

A laser light can be produced in different wavelengths such as ultraviolet ($100 \sim 400\text{nm}$), visible colors ($400 \sim 700\text{nm}$) and near infra-red ($700 \sim 1400\text{nm}$) and infra-red domains ($1400\text{nm} \sim 1\text{mm}$).

Laser impacts on combinational logic circuits are known to alter functioning. Current chip manufacturing technologies are in the nanometers range. This, and the laser's brief and precise reaction time, makes laser a particularly suitable fault injection means.

1.3.2 Photoelectric Effect of Laser on Silicon

SRAM (Static Random Access Memory) laser exposure is known to cause bit-flips [Skorobogatov 2003], [Darracq 2002], [Bar-El 2006], a phenomenon called *Single*

Table 1.1: Brief state-of-the-art of the most effective fault attacks on AES.

Title	Type	Target	Fault Model	Focalization	Nbr. of Distinct Faults	Nbr. of Faulty Realizations	Main Reference
Blömer and Seifert	SEA	Data (M_0)	Set or Reset	Bit	128	128	[Blömer 2003]
Chen and Yen	DFA	Data $7*(M_8)$ Key $4*(K_9)$	Random	Byte	11	22	[Chen 2003]
Giraud's bit	DFA	Data $16*(M_9)$	Bit-flip	Bit	16	~ 50	[Giraud 2005]
Giraud's byte	DFA	Key $4*(K_9)$ + $4*(K_{10})$ Data $4*(M_8)$	Random	Byte	12	250 for 14 bytes	[Giraud 2005]
Piret and Quisquater	DFA	Data $4*(M_9)$ or $1*(M_8)$	Random	Byte	4 or 1	8 or 2	[Piret 2003]
Choukri and Tunstall	RR	Round counter	Depending	Round counter value	1	2	[Choukri 2005]
Monnet and Al.	RR	Round counter	Bit-flip	1 or 2 bits	depends	depends	[Monnet 2006]
Park and al.	RR	Round counter	Depending	Byte	1	2	[Park 2011]
Dusart, Letourneux and Vivolo	SEA	Data $4*(M_9)$	Random	Byte	4	8	[Dusart 2003]
Robisson and Mannet	SEA	Data before R_1	Stuck-at	Bit	16	~ 256	[Robisson 2007]
Robisson and Mannet	SEA	Data at R_0	Stuck-at	Bit to byte	16	~ 256 to 4096	[Robisson 2007]
Kim and Quisquater	DFA	K_7	Random	Byte	1	2	[Kim 2008]
Tunstall, Mukhopadhyay and Ali	DFA plus Exhaustive	Data at R_8 input	Random	Byte	1	1	[Tunstall 2011]

Event Upset (SEU). By tuning the beam's energy level below a destructive threshold, the target will not suffer any permanent damage.

A conventional one-bit SRAM cell (figure 1.5) is made of two cross-coupled inverters. Every cell has two additional transistors controlling the cell's content access during write and read. As every inverter is made of two transistors, a SRAM cell contains six MOS.

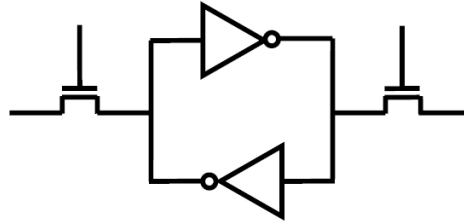


Figure 1.5: Architecture of a typical SRAM cell.

In each cell, the states of four transistors encode the stored value. By design, the cell admits only two stable states: a “0” or a “1”. In each stable state, two transistors are at an ON state and two others are OFF.

If a laser beam hits the drain/bulk reversed-biased P-N junctions of a blocked transistor, the beam's energy may create pairs of electrons and holes as the beam passes through the silicon. The charge carriers induced in the collection volume of the drain-substrate junction of the blocked transistor are collected and create a transient current that inverts logically the inverter's output voltage. This voltage inversion is in turn applied to the second inverter that switches to its opposite state: all in all, a bit flip happens [Darracq 2002], [Bar-El 2006].

From the opponent's perspective, an additional advantage of laser fault injection is *reproducibility*. Identical faults can be repeated by carefully tuning the laser's parameters and the target's operating conditions.

1.3.3 Different Parameters in a Fault Attack by Laser

In a laser attack, the opponent usually controls the beam's diameter, wavelength, amount of emitted energy, impact coordinates (attacked circuit part) and the exposure's duration. Sometimes, the opponent may also control the impact's moment¹, the target's clock frequency, V_{cc} and temperature. Finally, laser attacks may indifferently target the chip's front side or backside.

However, the chip's front side and backside have different characteristics when exposed to a laser beam:

¹i.e. the impact's synchrony with a given clock cycle of the target.

1.3.3.1 Front side attacks

The laser attacks on front side decapsulated chips are particularly suited to green wavelength ($\sim 532\text{nm}$). The visibility of the chip components makes positioning very easy in comparison to backside attacks. Nevertheless, because of the metallic interconnects' reflective effect, it is difficult to target a component with sufficient accuracy. In addition, sometimes the chip's layout has a mesh protection. On the other hand, progress in manufacturing technologies results in both a proliferation of metal interconnects and much smaller chips. Consequently, it becomes increasingly difficult to hit a target area.

1.3.3.2 Backside attacks

The silicon structure is a transparent area for infrared wavelengths. Because of this property, the backside laser attacks are more efficient, as the infrared rays ($\sim 1064\text{nm}$) enter deeply into the silicon and target components backwards. Positioning is more difficult for lack of visibility. Nonetheless, backside attacks allow the opponent to circumvent the reflective problem of metallic surfaces.

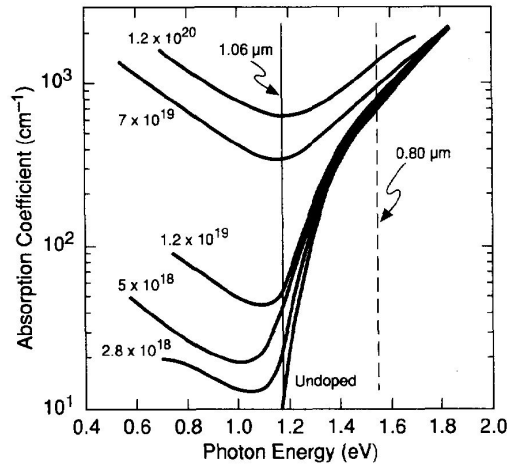


Figure 1.6: Absorption coefficient for silicon at various doping levels of p-type material.

The figure 1.6 shows the absorption coefficient for silicon at various doping levels of p-type material. In this figure, taken from [Melinger 1994], the photon energy is a function of laser wavelength.

In figure 1.6, the solid line and the dashed line mark the photon energies corresponding to the wavelengths $1.06\text{ }\mu\text{m}$ and $0.80\text{ }\mu\text{m}$ respectively.

According to figure 1.6 and [Melinger 1994], an infrared laser, e.g. a $\sim 1064\text{ }\mu\text{m}$ wavelength provides a deep penetration to the silicon layer and is suitable for backside laser fault injection on the integrated circuits. Besides, for a front side

attack, wavelengths lower than $0.80\text{ }\mu\text{m}$ are suitable as they don't enter deeply into the silicon. It includes a $\sim 532\text{ }\mu\text{m}$ green laser wavelength .

Before performing the laser fault injection, the target circuit must be prepared for such attacks. Depending on the equipment and the attack requirements, the decapsulation of the chip's front side or backside is needed.

In the next chapter, dedicated to the *security characterization*, we describe different stages of our laser experiments and circuit preparation.

Security Characterization

Contents

2.1	MicroPackS Security Characterization Laboratory	25
2.1.1	Laser Bench Characteristics	26
2.2	Circuit and Sample Preparation	27
2.2.1	Circuit Characteristics	27
2.2.2	Circuit Decapsulation	31
2.3	Security Characterization: First Mapping of Fault Injection Susceptibility	34
2.3.1	Laser Spot Parameters and Fault Injection	35
2.3.2	Exploration of our Circuit	37
2.3.3	Conclusions	39

Security characterization is a stage in the classical design flow of integrated circuits. Its utility is to verify the conformity of chip samples to the defined security specifications, from the beginning of the prototyping phase and through the mass production period.

Security characterization refers to the use of external techniques to examine the internal structure and properties of a circuit in order to evaluate its characteristics and resistance. A cryptographic circuit, e.g. a smart card, must have an acceptable level of security in order to achieve the four objectives of cryptography and to protect itself against any unauthorized interference. Analysis techniques are used to magnify any data leakage or any vulnerability of the circuit and also to evaluate its tamper-resistance.

2.1 MicroPackS Security Characterization Laboratory

MicroPackS security characterization laboratory is a part of a mutualized platform structure of the same name, created and shared by academic and industrial partners in the PACA region, France¹. Its main aim is to provide security characterization facilities for testing materials during the execution of R&D projects from the design phase through to the product validation phase.

¹<http://www.arcsis.org>

This laboratory is currently composed of five units: *Laser fault injection*, *Electrical tests*, SPA/DPA, EMA and *Contactless* characterization benches. Installation of the sixth unit for an *Advanced laser fault injection bench* is also planned for the near future.

As the main scope of this thesis concerns laser fault injection, we focus only on characteristics of the current laser bench.

2.1.1 Laser Bench Characteristics

The laser fault injection unit is composed of a laser bench, a control PC, an oscilloscope and other required equipment. The laser bench, shown in figures 8.4 and 8.5, is equipped with a YAG² laser emitter in three different wavelengths: 355 nm (ultraviolet), 532 nm (green) and 1064 nm (infrared). The target can be fixed on a programmable Prior Scientific *motorized stepper stage for upright microscopes* as an X-Y positioning table with 0.1 μm steps.

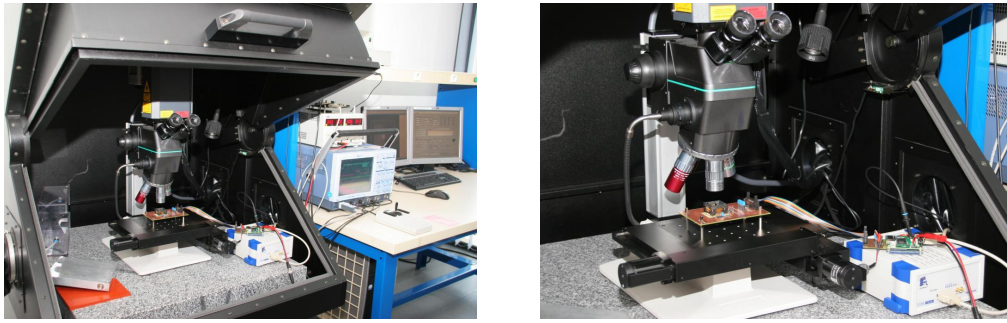


Figure 2.1: MicroPackS laserbench.

The nominal spot's shape is rectangular and controlled by a rectangular shutter opening. Each two parallel sides of the shutter can be opened between 0 and 2500 μm . As the beam passes through a choice of Mitutoyo lenses, it gets reduced by the lens' zoom factor and loses a significant part of its energy.

The X-Y positioning table, card reader, laser emitter and FPGA trigger board are connected via RS-232 to the control PC. All the parameters are controlled via SMART-I³ program. It is a running GUI⁴ developed under LabVIEW environment for the control PCs in different MicroPackS characterization benches. SMART-I has the possibility of sending commands to smart cards or other circuits under test and receiving and saving responses and signals. The FPGA trigger board receives an activation signal from the reader and sends a trigger signal to the laser after a delay defined on the SMART-I interface.

²Yttrium Aluminium Garnet or $\text{Y}_3\text{Al}_5\text{O}_{12}$

³Secured Multi-characterisation Test Interface

⁴Graphical User Interface

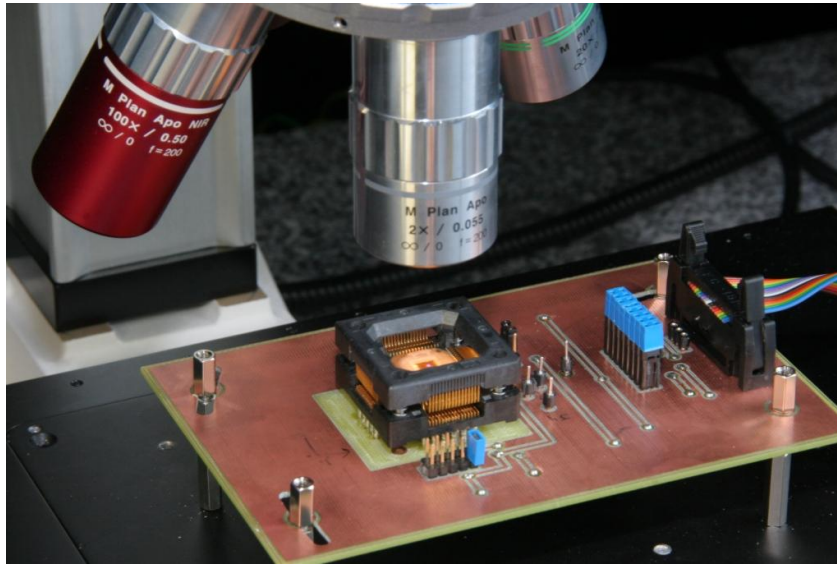


Figure 2.2: Close-up on our circuit at the laserbench.

The laser pulse duration is about 5 ns. A waiting time of about 200 μ s with a tolerance of 500 ns is required for laser internal triggering, switching and heating between each laser command and its emission. Nevertheless, successive laser emissions can be done with a frequency of 50 Hz.

2.2 Circuit and Sample Preparation

For most of our characterization tests in this thesis, we used a smart card emulation board built in the laboratory. It is composed of an 8-bit 0.35 μ m 16 MHz RISC microcontroller with integrated 128 KB flash program memory, 4KB EEPROM and 4KB SRAM.

The microcontroller is placed on the circuit via a ZIF support that allows the chips to be easily changed for different tests. A hole at the center of the chip support (and the circuit board) permits a physical access to the chip's backside for laser experiments.

The device runs the *Simple Operating System for Smartcard Education* called SOSSE [Bruestle 2002] for simulating the smart card environment.

2.2.1 Circuit Characteristics

SOSSE is an open source operating system and conforms to the principal ISO/IEC 7816 standard commands for smart cards. It was developed as an open source project under the Gnu Public License [Bruestle 2002]. SOSSE is mainly programmed in ANSI C language.

SOSSE supports the T=0 communication protocol between card reader and smart card [Bruestle 2002]. The T=0 protocol is the most widely used communication protocol and was standardized in ISO/IEC 7816-3 and 7816-4. It was designed in the early years of smart card technology for a minimum memory usage and a maximum simplicity. It is used by smart cards in Global System for Mobile communications (GSM) [Rankl 2003]. The T=0 is a byte-oriented, asynchronous and half-duplex protocol [Mayes 2008]. Therefore, the smallest unit processed by the protocol is a single byte.

The smart card always functions as a slave for the card reader. The smart card receives a command from the card reader as a master, then the smart card microcontroller executes corresponding operations and sends a response back to the card reader. The structures of transmission commands and responses in the T=0 protocol are defined by [ISO/IEC 7816-3 2003].

A T=0 protocol transmission command is sent by the reader to the card and contains a mandatory 4-byte header plus an optional 5th byte header and optional data. The containing header is composed of a class byte (*i.e.* CLA), an instruction or command byte (*i.e.* INS) and two, or optionally, three parameter bytes (*i.e.* P1, P2 and the optional P3). The header is followed by the data section that has from 0 to 255 bytes of data.

A T=0 protocol transmission response is sent by the card to the reader and contains from 0 to 256 bytes of data followed by a mandatory 2-byte status word.

The length information of the command data or response data is provided only by parameter P3 in ISO/IEC 7816-3. Figure 2.3 shows the structure of a command in T=0 protocol, according to [Rankl 2003].

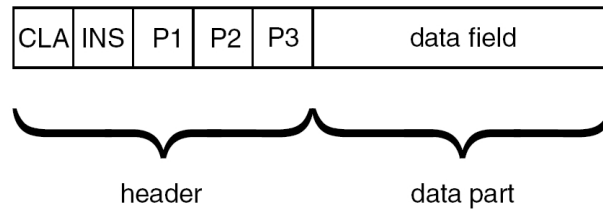


Figure 2.3: The structure of a command in T=0 protocol.

In order to program our circuit with SOSSE, the source code must be compiled. The compilation of SOSSE creates two binary files for loading onto the circuit's flash and EEPROM via linking cable and program.

This microcontroller communicates with the card reader at 3.59 MHz frequency. In SOSSE environment, similarly to smart card ISO/IEC 7816 standard, the commands are sent as a T=0 protocol transmission command to the smart card.

For performing our tests, we used a customized version of SOSSE modified in our laboratory by adding a set of new commands:

1. **Read_RAM**: Reading a part of SRAM contents, from 1 to 256 bytes from a given start address. For instance: In the command below, 0C refers to the **Read_RAM** command. 0A 00 refers to the start address and 10 determines the length of requested bytes in hexadecimal values.

Fct : Cmd TO Cmd : 80 0C 0A 00 10

Then, the 16 bytes in the response are the requested data. The status, equal to 90 00, indicates that the process has been completed successfully.

Rsp : 2B 7E 15 16 28 AE D2 A6 AB F7 15 88 09 CF 4F 3C
Status : 90 00

2. **Write_RAM**: Storing a given byte value on all the bytes of SRAM between addresses 0x0500 and 0x10FF. In the given example below, the command 08 refers to **Write_RAM** and the final byte, equal to FF, determines the value that will be written on the addresses from 0x0500 to 0x10FF.

Fct : Cmd TO Cmd : 80 08 00 00 01 FF

Then, the status 90 00 indicates that the process has been completed successfully.

Rsp : Status : 90 00

3. **AES_Encrypt**: This is the command for feeding-in a plaintext, in order to encrypt it with the embedded AES. The appropriate command is defined as 40 followed by the length of plaintext. For example, the below command contains 10 to determine that the plaintext length is 16 bytes. The command is followed by a plaintext equal to 32 43 .. 34 in order to be encrypted by AES:

Fct : Cmd TO Cmd : 80 40 00 00 10 32 43 F6 A8 88 5A 30
8D 31 31 98 A2 E0 37 07 34

Then, the card answers the following status. First status byte equal to 61 indicates a normal processing. The hexadecimal value of 10 on the second byte shows the number of remaining bytes. It corresponds to the ciphertext length.

Rsp : Status : 61 10

Therefore, we need a second command in order to read the ciphertext.

4. **Read_Ciphertext**: This command is added for retrieving the ciphertext after AES encryption. The command C0 asks the remaining bytes for the answer from the card. The value equal to 10 refers to the number of expected bytes in the answer:

Fct : Cmd TO Cmd : 80 C0 00 00 10

Then the card response contains the ciphertext. The status equal to 90 00 means no further qualification:

```
Rsp : 39 25 84 1D 02 DC 09 FB DC 11 85 97 19 6A 0B 32
Status : 90 00
```

According to the implementation of SOSSE, after each circuit reset, all the variables are copied to SRAM. We added an embedded AES encryption function to our SOSSE implementation. It operates by the `AES_Encrypt` command. Each plaintext is sent as a parameter in `AES_Encrypt` command. The corresponding ciphertext is retrieved by the `Read_Ciphertext` command.

In our implementation, the secret key K for AES is embedded in the code. After each circuit reset, the round keys are calculated and stored in the microcontroller's SRAM. The S-Box look-up table is also included in the program code and, after each reset, it is copied to SRAM. The encryption process refers to the stored round key values and S-Box look-up table in SRAM. Figure 8.6 shows an overview of our AES implementation and its principle operations.

Our embedded AES encryption function permits the selection of the AES-128, 192 or 256 bit version, in 10, 12 or 14 rounds respectively. In our experiments, reported in this thesis, we used only the AES-128 version.

As underlined in the next part of this thesis, the main entry point for fault attacks is the storage of sensitive data in the SRAM.

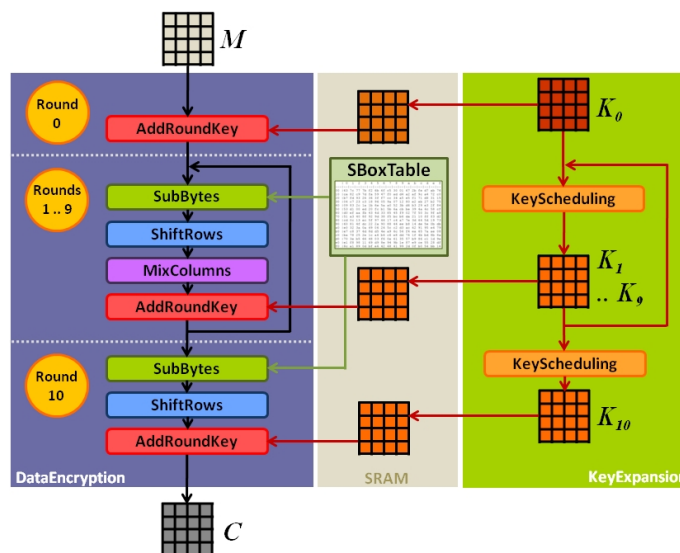


Figure 2.4: An overview of our AES implementation.

For our tests requirements, we often applied other specific changes, such as adding other new commands, developing triggers on the chip's I/O ports or modifying the actual command parameters to our customized SOSSE.

Hereafter, we use only the terms, “our chip”, “our circuit” or “our microcontroller” to refer to the aforementioned microcontroller, used for almost all the experiments, except when another circuit is mentioned.

For the tests done before countermeasure implementations described in chapter 5, our circuit did not have any software or hardware countermeasures.

2.2.2 Circuit Decapsulation

Before performing the laser fault injection, the target circuit must be prepared for such an experiment. Depending on the equipment and the attack requirements, the front side or the backside of the circuit can be decapsulated.

2.2.2.1 Front Side Decapsulation



Figure 2.5: Chemical etching bench in MicroPackS laboratory and close-up on Nisene JetEtch II decapsulator.

The chemical etching is more suitable for a front side chip decapsulation. It can preserve the circuit integrity and functionality when it is performed in a proper manner. The chemical decapsulation can be performed manually by the opponent or semi-automatically or automatically by decapsulation machines. The encapsulant must be removed while preserving the integrity of the die, bond pads, bond wires and lead frame interconnects.

MicroPackS platform is equipped by a Nisene JetEtch II decapsulator, for front side chip decapsulation. This automated and programmable acid etching system, shown in the figures 2.5, uses fuming sulfuric acid (oleum) and fuming nitric acid. The machine offers the possibility of using pure or mixed ratios of these acids at the desired temperature, between 20 and 250°C.

The optimal package opening of modern chips usually requires the use of mixed etchants. For the careful reproducible opening of modern chip packages, particularly those constructed with copper lead frames, or other copper components, exact mix ratios of acids is needed. It is not only required to preserve structural components, but also to maintain electrical functionality [Nisene].

The chemical chip etching is a very sensitive process. A small degree change in the chemical decapsulation temperature can make a notable difference in the final result. It either creates a perfect decapsulated chip that keeps all its functionalities or it creates a deep corrosion and disruption of critical structures, such as die coats.

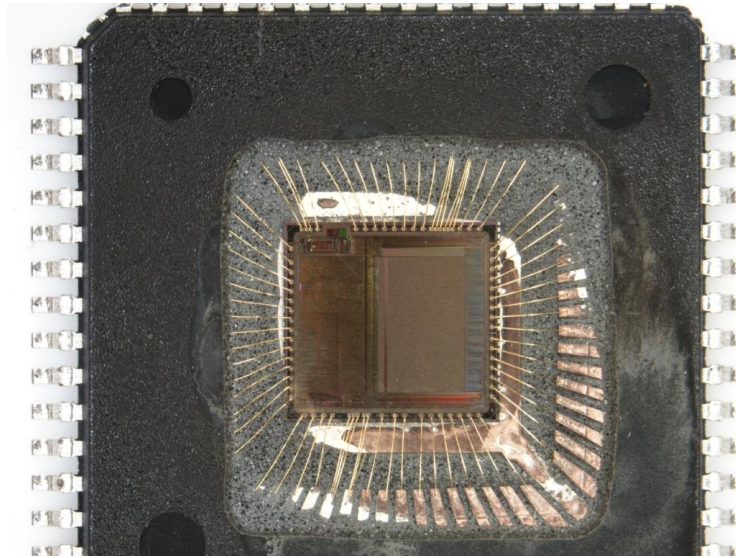


Figure 2.6: A front side decapsulated sample of our chip by chemical etching.

2.2.2.2 Backside Decapsulation

For a backside laser attack, mechanical decapsulation is more suitable than chemical methods. It can better maintain the scratching distance on the silicon layer and thus the remaining layer is better furnished. For a backside decapsulated chip, a thinned and perfectly polished surface is required for an accurate laser attack.

The backside decapsulation bench at MicroPackS laboratory is equipped with an Ultra Tec ASAP⁵-1 decapsulator unit, as shown in figure 2.7 for backside chip decapsulation. This automated mechanical machine is formed by a rotating part which contains different scratching and polishing tools, controlled in the z-direction [Ultra Tec]. The chip sample is fixed on an X-Y oscillating table and the rotating part, depending on which tool is mounted, can scratch or polish the backside surface.

⁵Automated Selected Area Preparation system

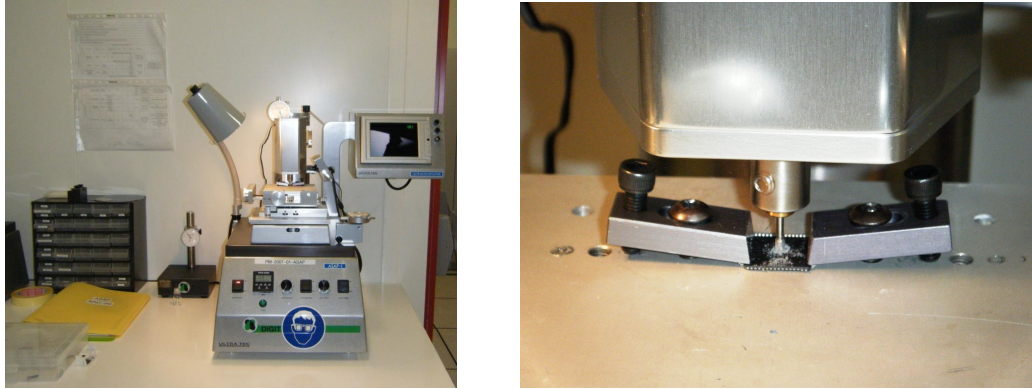


Figure 2.7: Mechanical etching bench in MicroPackS laboratory equipped with an Ultra Tec ASAP-1 decapsulator.

By using different tools in conjunction with loose or fixed abrasives, we obtain a mirror-like surface without any scratches.

Contrary to the chemical front side automated etching that takes a few minutes, from the beginning to the end, the mechanical decapsulation requires different steps and takes between thirty minutes and a few hours.

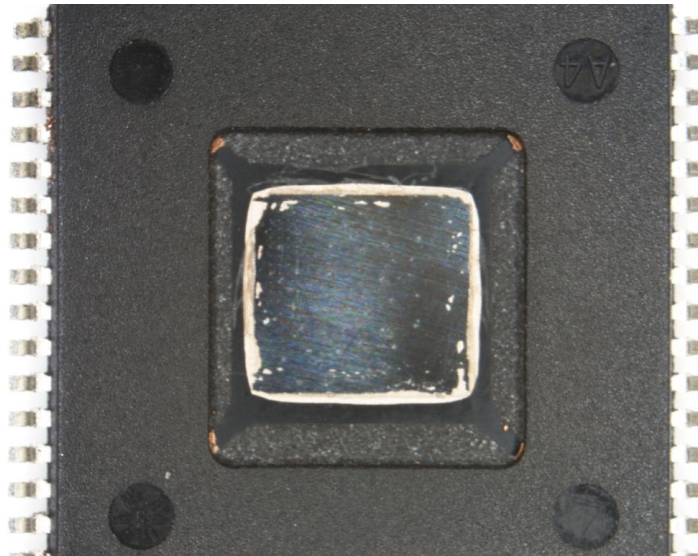


Figure 2.8: A backside decapsulated sample of our chip by mechanical etching.

After the decapsulation step, performing some tests and verifications is necessary prior to laser fault injection attacks in order to guarantee the correct functioning of the circuit.

In this section, we describe chip decapsulation and preparation for laser fault

injection. In the next chapter, we will show our practical results for fault attacks on a decapsulated microcontroller that embeds an AES.

2.3 Security Characterization: First Mapping of Fault Injection Susceptibility

Before starting our fault attacks experiments on AES, a good familiarity with our circuit, as well as the impact of laser emissions was necessary.

By magnifying the front side decapsulated samples of our microcontroller, we could identify its different components according to their forms and also according to the given data in its datasheet.

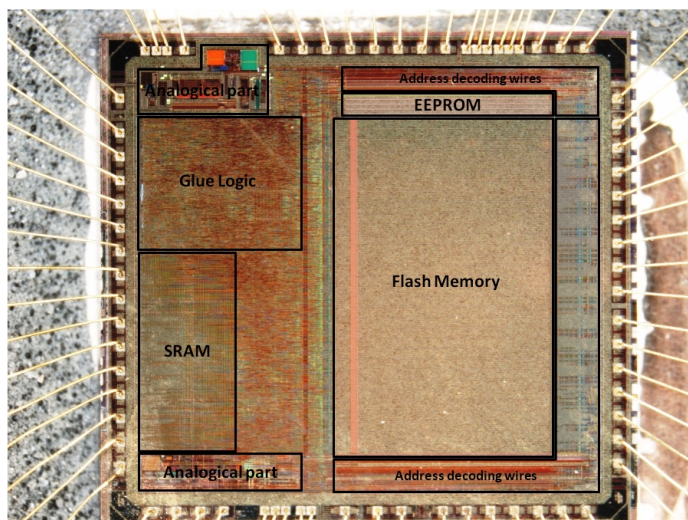


Figure 2.9: Main identified components on the original top metal overview photo.

Figure 2.9 shows the mapping of the main functional blocs of the chip, *i.e.* SRAM, flash memory, EEPROM, the analogical parts and the logic of our circuit. In configuration many variables are stored in the SRAM; therefore, detailed information about its organization is necessary. Moreover, according to the section 1.3, laser impact on the SRAM structure is SEU when its energy is lower than the destructive threshold. So, this experiment permits us to measure the effects of laser impact on our circuit.

In our circuit and AES implementation, the SRAM has an important role, because all the round keys, the `SubBytes` look-up table, the round counter and many variables are stored on it. Besides, as we described in 1.1.5, SRAM is a perfect target for laser fault injection. Therefore, we studied our circuit's SRAM with particular attention.

For a full cartography of our circuit's SRAM, we used a modified version of SOSSE with only `Read_RAM` and `Write_RAM` commands. We moved stack and data sections in the memory to the beginning and the end corners of the SRAM for finding the spatial coordinates of all the cell blocks.

We modified `Write_RAM` command in order to write on the SRAM addresses except 80 bytes reserved for stack and data sections. We performed a cartography of the SRAM by a full scan of memory by laser emission.

By using these two commands, we could automatically perform this procedure for any scanned area on the SRAM:

1. Writing SRAM cells with a fixed value, for instance `0x00`.
2. Performing a laser emission.
3. Reading SRAM cells. Finding all the differences between the current and the written value. Allocating the address of changed bytes to the spatial coordinates of the scanned area.
4. Going to the next scanning area and repeating this procedure, except if the scanning area is finished.

By using surgical laser fault injection on the SRAM cells and this procedure, we could achieve the figure 2.10 as the cartography of SRAM addresses.

In our AES implementation, the `SubBytes` look-up table, the round keys and some variables were stored in SRAM. So, this cartography was very useful for our upcoming experiments.

2.3.1 Laser Spot Parameters and Fault Injection

In our experiment, we surveyed the relationship between spot parameters and the number of injected faults. When a laser spot becomes larger, it encounters more memory cells and the number of injected faults increases. In addition, the laser energy is involved in the number of injected faults. Table 2.1 shows that faulty bits increase according to the growth of laser energy.

Table 2.1: Effects of laser energy on the number of injected faults into the SRAM using a spot size of $3.75 \mu m \times 3.75 \mu m$. The experiment is done on a front side decapsulated sample of our chip.

Energy	180 <i>pJ</i>	1.8 <i>nJ</i>	9 <i>nJ</i>	18 <i>nJ</i>
Average fault number	14.3	60.8	80.4	87.3

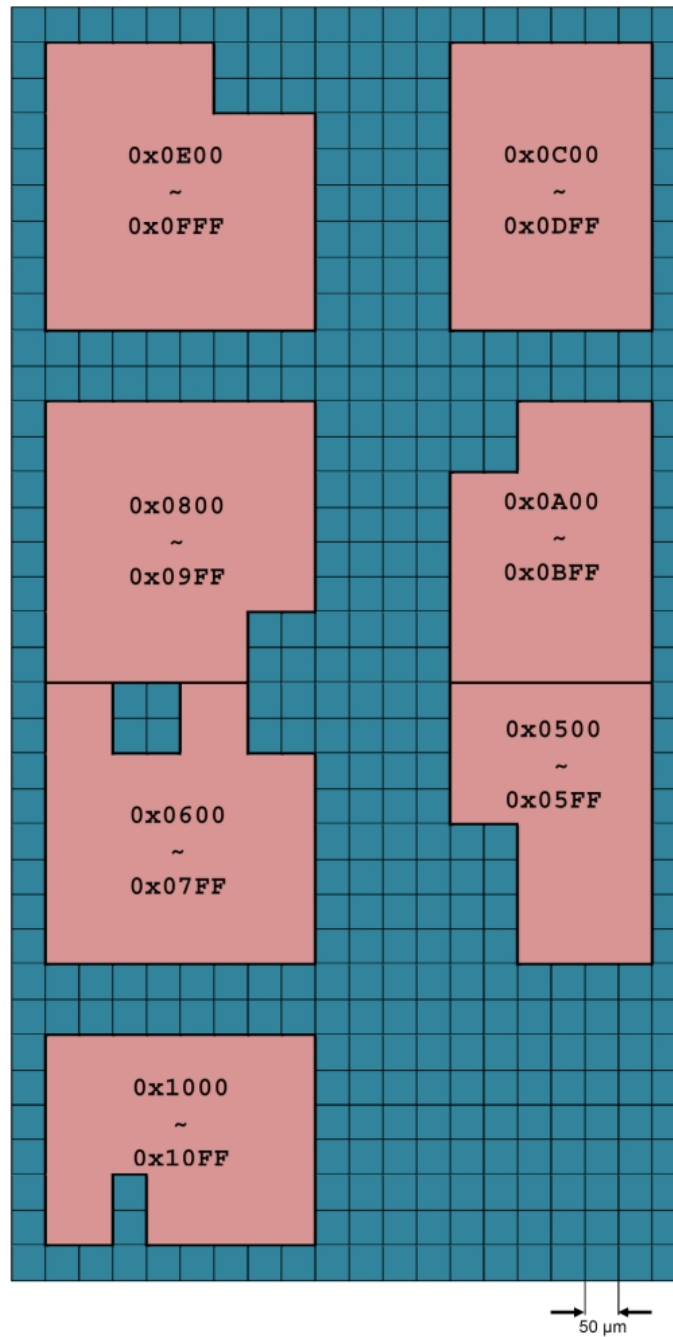


Figure 2.10: Physical allocation of the SRAM's bytes on the front side of our decapsulated microcontroller, discovered by laser fault injection.

2.3.1.1 Single-Bit and Multiple-Bit Fault Injection

The number of faulty bits on each byte was another parameter surveyed during our preliminary experiments. We found that, when the laser beam width is smaller than $\simeq 44\mu m$, all the faulty bytes on our circuit contain a single-bit fault. With larger laser beams, the chance of multiple-bit faults increases. However, we cannot create more than three faulty bits on each byte, when the beam width is smaller than $93\mu m$. In addition, between faulty bytes, the frequency of double-bit and triple-bit faults remains very low.

Our results show that the chance of injecting faults on same-value bits of several bytes is much greater than on other bits of the same bytes. It means that in several physical implementations of SRAM (e.g. in our microcontroller or also in the microcontroller described in [Skorobogatov 2003]), the bits of a same value are designed and built close together for a block of bytes in the memory array. In these implementations, the distance of two bit cells of the same value in a block of bytes (e.g. 256 bytes) is usually much closer than the distance of a bit to its neighboring bits of the same byte. So, in the case of a surgical laser fault injection, the chance of creating single-bit faults is very high.

2.3.2 Exploration of our Circuit

In addition to the fault injection experiments on the SRAM, we performed a full exploration of our circuit by laser emission and we analyzed the circuit behavior.

Our experiments were done twice: In the first experiment, we read all the SRAM content before and after laser emissions to detect any changes on it. We also performed a second laser experiment on different areas of the chip during an AES encryption, in order to detect any effect on the encryption process.

We could classify the effects of laser emission on our chip into four categories:

1. No effect: We could not detect any fault on SRAM, AES encryption or circuit behavior after laser emission.
2. Faulty response with correct functioning: We could inject faults by laser emission on stored values in SRAM, communicated values or calculations. The circuit maintains its correct functioning after laser emission.
3. Temporary circuit error: Sometimes, laser emission created temporary faulty behavior. In these cases, our circuit did not answer the first reader command after laser emission, but it recovered its correct functioning very shortly thereafter.

For instance, in the below example, we targeted AES round keys stored in the SRAM during encryption. The laser emission is carried out after sending the first command and before receiving the corresponding status from the reader.

```
Fct : Cmd T0 Cmd : 80 40 00 00 10 01 23 45 67 89 AB CD
EF 01 23 45 67 89 AB CD EF Rsp : Status : 61 10
```

```
Fct : Cmd T0 Cmd : 80 C0 00 00 10 Rsp : Status : 6D
00
```

```
Fct : Cmd T0 Cmd : 80 0C 0C 2D 10 Rsp : 2B 7E 15 16 28
AE 92 A6 AB F7 15 88 09 8F 0F 3C Status : 90 00
```

The card correctly sends a status equal to 61 00 at the end of the first command. Therefore, its answer to the second command is null, with a status equal to 6D 00. This means “*command not supported*”. Then, the circuit recovers its correct functioning and correctly answers the following command, even if faulty values are injected on the SRAM contents.

4. Permanent circuit error: Sometimes, laser emission created a permanent faulty behavior. In these cases, our circuit did not answer any command sent by the reader after laser emission. Waiting for several seconds or minutes did not recover the original card functioning. In these cases, a circuit *cold reset* was necessary, *i.e.* turning off the supply voltage and turning it on again or disconnecting and reconnecting the circuit from the reader. Therefore, the effect of such faults were permanent, but not destructive.

In another experiment, we looked at the research of our colleagues in laser fault injection on a similar circuit during AES encryption. They reported a successful attack by targeting a part of the circuit, as shown in figure 2.11. The specified location corresponds to a set of wire lines implemented in the flash memory area.

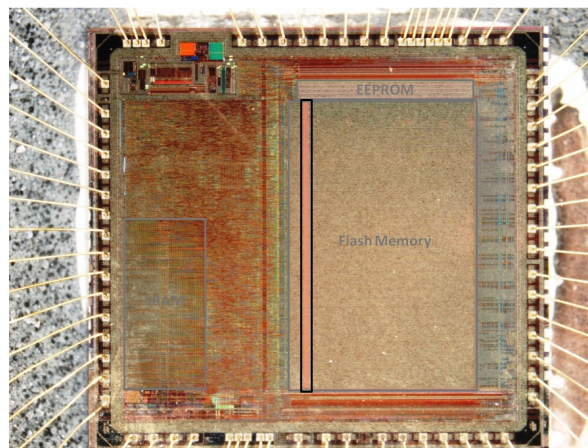


Figure 2.11: The vertical wire lines in the middle of our circuit.

They guessed that in their experiment they attacked the **SubBytes** look-up table in the flash memory by targeting the circuit area shown in figure 2.11. Therefore, we tried to reproduce the same attack and we performed it successfully in our circuit.

In the next step, we tried to discover the reason for the aforementioned attack. In our implementation, the **SubBytes** look-up table is copied after reset from flash memory to SRAM. So, the copy version of the **SubBytes** look-up table must be used during the encryption. In order to confirm that the change did not come from faulty flash memory contents, we added a SOSSE command to change only one byte value of the **SubBytes** look-up table in SRAM. We changed the corresponding value to the used **SubBytes** look-up table values for the temporary ciphertext at the beginning of 10th round (according to our chosen plaintext and to the used key). We found that this change results in a faulty ciphertext; but, when we rewrite the original value, the next encryption is done correctly.

In addition, we surveyed the effect of laser emission time on the faulty value and discovered that the faults happen on the temporary ciphertext and therefore, the wire lines must correspond to the data bus.

2.3.3 Conclusions

We obtained an understanding of laser parameters for injecting faults on our circuit during our preliminary experiments. In the next chapter, we use this knowledge for producing the well-known DFA attacks on our circuit.

Practical DFA by Laser on the AES

Contents

3.1	Problematic in Practical DFA by Laser	42
3.2	Piret-Quisquater's Differential Fault Analysis	43
3.2.1	Attack Scheme	43
3.2.2	Practical Experiment	44
3.2.3	Practical Experiment Without Memory Access	48
3.2.4	Conclusion	49
3.3	Giraud's Single-Bit Differential Fault Analysis	51
3.3.1	Attack Scheme	51
3.3.2	Practical Experiment	53
3.3.3	Conclusion	55
3.4	An Extended Single-Bit DFA for Multiple-Byte Faults . . .	55
3.4.1	1 st Scheme for the Extended Attack	56
3.4.2	2 nd Scheme for the Extended Attack	57
3.4.3	Practical Experiment	58
3.4.4	Conclusion	59
3.5	Feasibility Comparison of Previous Attacks	60
3.5.1	Review of Piret-Quisquater's Experiment	60
3.5.2	Review of Giraud's Bit Experiment	64
3.5.3	Conclusion	65
3.6	An Extended Multiple-Byte DFA	68
3.6.1	Attack Scheme	68
3.6.2	Practical Experiment	71
3.6.3	A More Sophisticated Practical Experiment	72
3.6.4	Conclusion	74
3.7	Conclusion	74

In chapter 1, we reviewed the main fault attack techniques. Among them, laser emission is a fault injection means with good localization on the circuit. We know that it is possible to inject faults on the cryptosystems by laser emission. In our

preliminary tests, we also proved the possibility of injecting laser faults within our circuit. We determined that it is possible to change some byte values by laser emission on SRAM. Now, the main problem is to verify whether appropriated faults can be injected by laser in order to break an AES cryptosystem. In this chapter, our aim is to investigate the possibility of producing the required fault described in theoretical DFA attacks on the AES, such as [Piret 2003] and [Giraud 2005].

3.1 Problematic in Practical DFA by Laser

The theoretical DFA attacks on the AES are dependant on single-byte or single-bit fault models. The minimal diameter of a laser spot cannot successfully decrease to smaller than $\varnothing 1\mu\text{m}$, due to optical diffraction. Besides, as the technology advances, the number of transistors grows on the incident area of a $\varnothing 1\mu\text{m}$ spot.

Figure 3.1 shows a comparison between a $\varnothing 1$ and a $\varnothing 10\mu\text{m}$ laser spots and a SRAM cell in different technology scaling. It also shows that a $\varnothing 1\mu\text{m}$ spot may have a bigger effective area on the chip which depends on the laser energy level. This minimal beam hits several transistors on the most advanced technologies and cannot be physically limited, such that it only targets a single-bit/byte. Besides, a laser equipment providing with a $\varnothing 1\mu\text{m}$ spot is very expensive and not accessible to most opponents. So, injecting single-bit/byte faults or even a few byte faults needs more accurate equipment and becomes less feasible using cheap laser facilities.

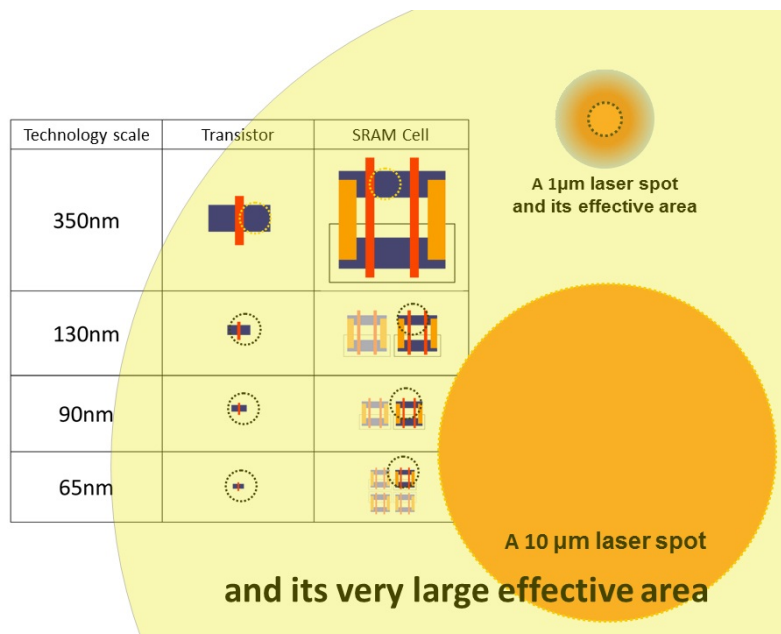


Figure 3.1: $1\mu\text{m}$ & $10\mu\text{m}$ laser spot diameters vs. technology scaling.

In the following sections of this chapter, we describe two classical DFA methods

on AES. Our experiments require single-bit or single-byte laser fault injection. We study the possibility of injecting required faults by our laser equipment on our circuit.

3.2 Piret-Quisquater's Differential Fault Analysis

3.2.1 Attack Scheme

Gilles Piret and Jean-Jacques Quisquater presented in [Piret 2003] a theoretical DFA attack on AES. In order to be successful, this attack requires the injection of a single-byte fault into the temporary ciphertext between the MixColumns output of the antepenultimate round and the MixColumns input of the penultimate round.

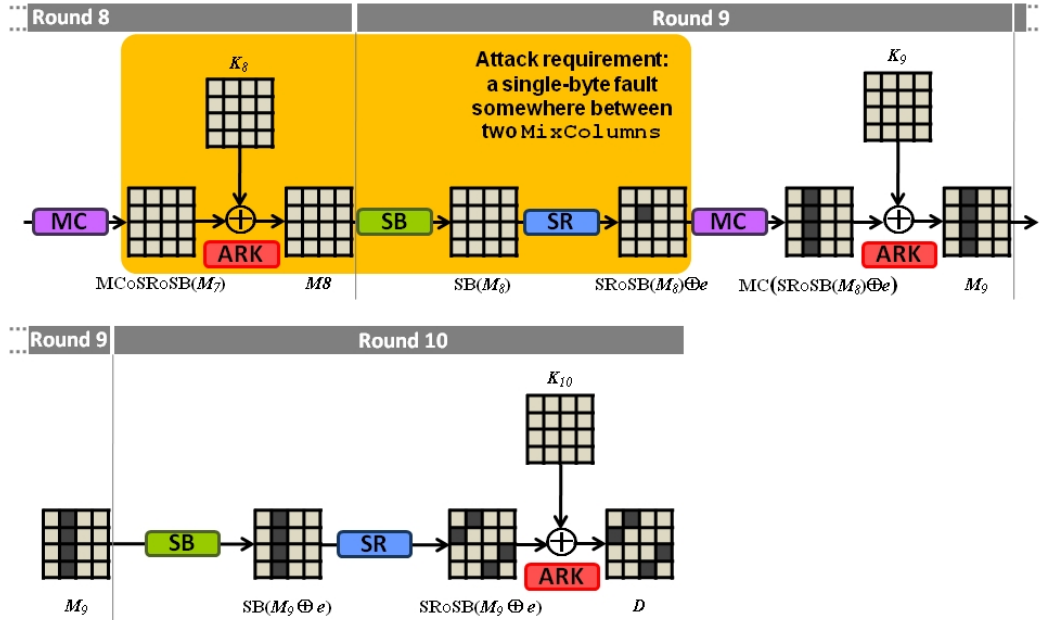


Figure 3.2: Propagation of a single-byte fault, injected at MixColumns input of 9th round.

Figure 3.2 shows a tuning window that extends between MixColumns exit of round 8 and MixColumns entry of round 9 for a single-byte fault injection according to attack requirement. Figure 3.2 also presents the fault propagation and diffusion into four bytes of the final ciphertext. Thus, the attack scheme described in [Piret 2003] allows the opponent to infer some information on the four corresponding bytes of K_{10} by processing the correct and faulty ciphertexts and checking over the list of all the related possible single-byte faults.

By repeating this process twice (i.e. by iterating the attack for a different plaintext) the exact value of the four bytes of K_{10} is found with a success rate of about

98% [Piret 2003]. The procedure is repeated to target K_{10} 's remaining bytes. Finally, $K = K_0$ is inferred by reversing the key expansion operations.

3.2.2 Practical Experiment

Now, we examine how we can implement this attack on our circuit which embeds an AES. We described in section 2.3 that in our circuit the round keys are derived and stored in the microcontroller's SRAM. Since it is very difficult to target the chip's ALU¹ and to inject only a single-byte fault during a very specific instant, between the end of MixColumns of the 8th round and before the MixColumns of the 9th round, targeting K_8 is an easier and also, a more accurate option for our attack.

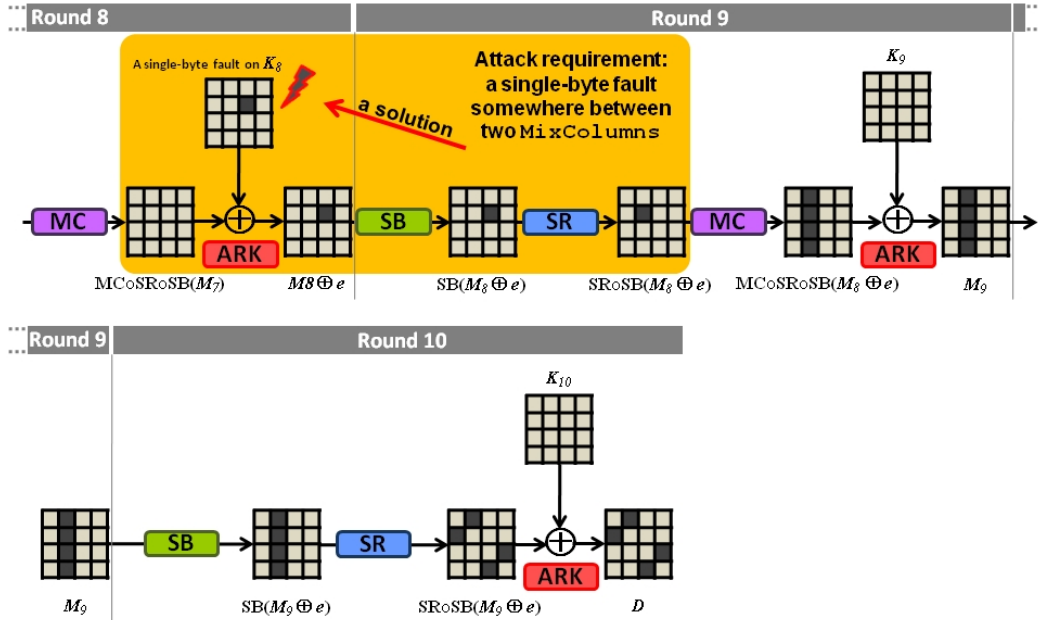


Figure 3.3: Effects of a single-byte fault at SubBytes input of 9th round.

The figure 3.3 shows how a single-byte fault on the antepenultimate round key can satisfy the attack requirement. For discovering each quadruple of bytes on K_{10} , a single-byte must be injected on one of corresponding 4 bytes on K_8 during the encryption of the first text. Then, a second fault injection must be performed during the encryption of the second text. Therefore, the attack scheme allows the opponent to reveal all 4 bytes of K_{10} .

For each of the 3 remaining quadruple of K_{10} bytes, the attack must be repeated by targeting one of the corresponding K_8 bytes.

According to the results and the spatial coordinates of the round keys obtained during our preliminary tests described in section 2.3, we tried to inject the first

¹Arithmetic Logic Unit

single-byte fault on K_8 in order to reveal the first four bytes of K_{10} , after two experiments.

The command `Read_RAM` described in section 2.2 helped us to read and detect all the faults injected on the round keys after each laser emission. Despite fine-grained energy and spatial control we detected faults in keys neighboring K_8 .

To overcome this problem, we used a three step solution:

1. **Spatial positioning:** We searched the spatial coordinates for laser fault injection on the first single K_8 byte, even if neighboring round keys were affected by faults.
2. **Refining spatial coordinates:** We tried to maintain fault injection on a single-byte of K_8 . In order to exclude any fault on the following round keys (i.e. K_9 and K_{10}), we performed micrometer displacements of the laser spot on the circuit using the positioning table. In this step, we did not care about any injected fault on the previous round keys. So, we isolated K_9 and K_{10} from any fault.

There is a solution for checking spatial coordinates: Performing a laser emission in the short time interval after the use of K_8 and before the use of K_9 . Therefore, the first encryption will be correct and the following ones will become faulty until the next circuit reset.

3. **Temporal positioning:** We established temporal accuracy of the laser emission time. We defined it in a short interval time after `AddRoundKey` of the 7th round and before `AddRoundKey` of the 8th round. Consequently, the logical effect of any fault in the previous round keys was discarded, despite the fact that the faults were injected and physically were present on the SRAM. So, the single-byte fault on K_8 was the only faulty round key byte that participated to the encryption.

Figure 3.4 shows how we could confine faults to a single byte of K_8 . We obtained single-byte faults by controlling the laser's spatial localization and shooting time. Note that figure 3.4 is obviously just a model of the SRAM's architecture. It describes our technique and does not correspond to real address allocation.

In our experiments, we successfully injected faults into 4 distinct bytes of K_8 that reveal all of the 16 bytes of K_{10} . Fault injection on only 4 bytes of K_8 for two different plaintexts sufficed to release information on all the 16 bytes of the key and to run Piret & Quisquater's attack. Our experiments were conducted on our chip's front side, with a $20\times$ lens, a green laser spot of $3.75 \times 3.75 \mu m$ and $\simeq 15pJ$ energy per shot (at laser emitter exit).

As shown in the topmost part of figure 3.4, we searched K_8 's precise storage area by monitoring the number and the type of faults in the ciphertext. Then (middle part of figure 3.4), by a precise beam localization, we managed to ensure that the

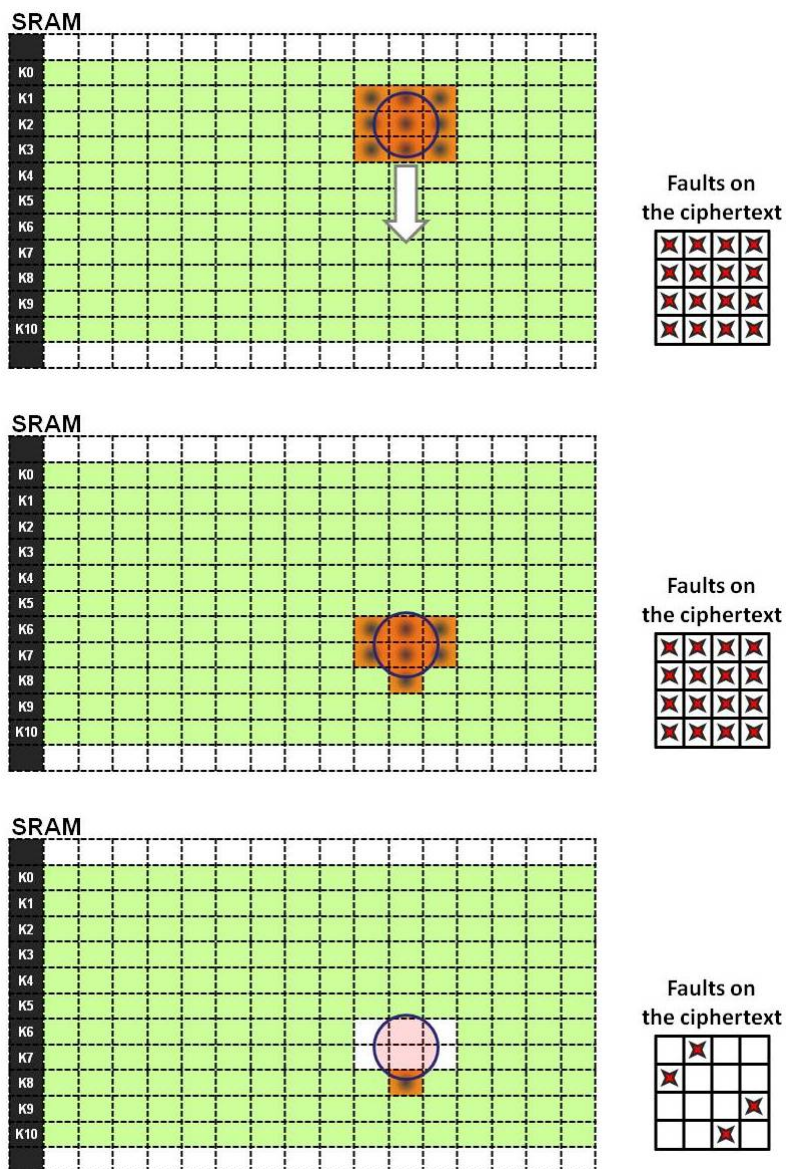


Figure 3.4: The exploration processes of our experiment are shown in three steps. First, we searched for K_8 bytes in SRAM. Then, we displaced the spot on SRAM in order to target only one K_8 byte and nothing on the following round keys. Finally, by adjusting precisely the timing of laser emission on t_8 , we logically injected only a single-byte fault on K_8 .

fault injection to only corrupts one byte of K_8 . This, however, did not turn out to be fully deterministic as sometimes we would also inflict faults to previous round keys. At that point (lowermost part of figure 3.4), by fine-tuning spatial and temporal beam localization (just after the use of K_7), we could restrict the injecting faulty bytes only to K_8 . This is the exact assumption of Piret-Quisquater's scenario.

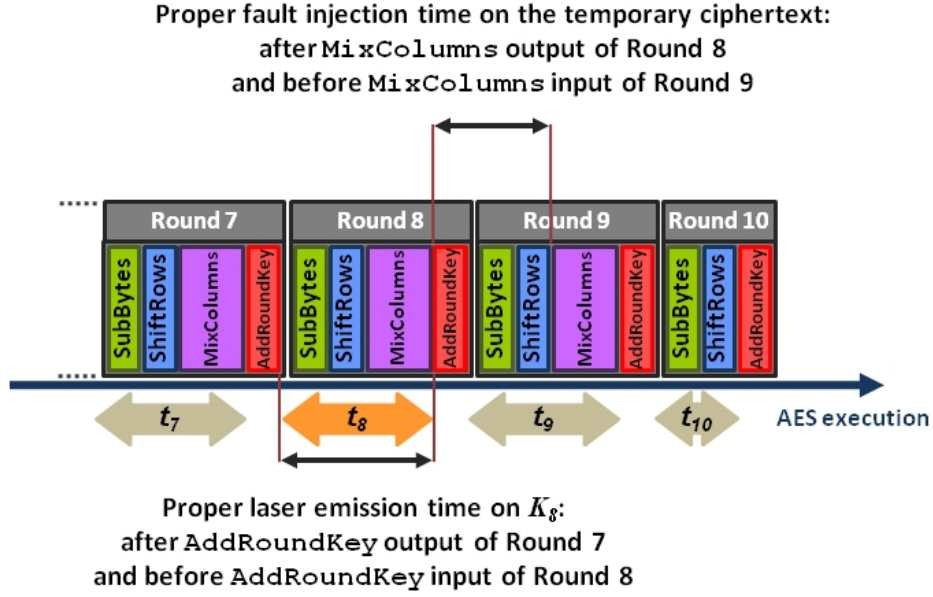


Figure 3.5: Piret-Quisquater's attack timing.

Indeed, between **AddRoundKey** operations of two consecutive rounds, there is a time interval for injecting faults that effects only logically the oncoming round keys. Even, if they change previous round keys, their faulty values don't participate in the encryption, despite the fact that they are physically present. Hereafter, we name each of these time intervals according to its respective round. For instance, we call t_8 , the time interval for injecting faults between the 7th round and the 8th round **AddRoundKey** operations. Figure 3.5 shows these time intervals and highlights the corresponding one for performing our Piret-Quisquater's DFA experiment.

In summary, we could successfully perform Piret-Quisquater's DFA in our circuit by using a laser beam that hits several bytes. In conclusion, that careful spatial and temporal coordination may allow the attacker to deceive the encryption process by logically considering only a single-byte fault that corresponds to Piret-Quisquater's scheme. So, an accurate X-Y positioning table may compensate the extra size of a laser beam that hits several bytes.

However, this experiment is done on our circuit that permits an access to its memory for reading the injected faults. Now, the next question is: Is it still possible to achieve this fault injection successfully without memory access?

3.2.3 Practical Experiment Without Memory Access

We surveyed the feasibility of the previous experiment, regardless of having any access to the memory. Even, without reading the SRAM contents after laser emissions, we successfully performed the experiment.

For this new experiment, we blindly searched the coordinates of corresponding SRAM bytes on the circuit just by monitoring the final ciphertext. When any fault is not injected on the round keys, the final ciphertext did not have any fault. Contrarily, if any fault is injected on any round round key prior to K_8 , the final ciphertext is obtained fully faulty. Because, such a single-byte fault on K_7 or on previous round keys passes at least two times into consecutive **ShiftRows** and **MixColumns** operations that amplify the fault and thus, corrupt all the bytes of the ciphertext. Figure 3.6 shows the consequences of one single-byte fault injected into any round key.

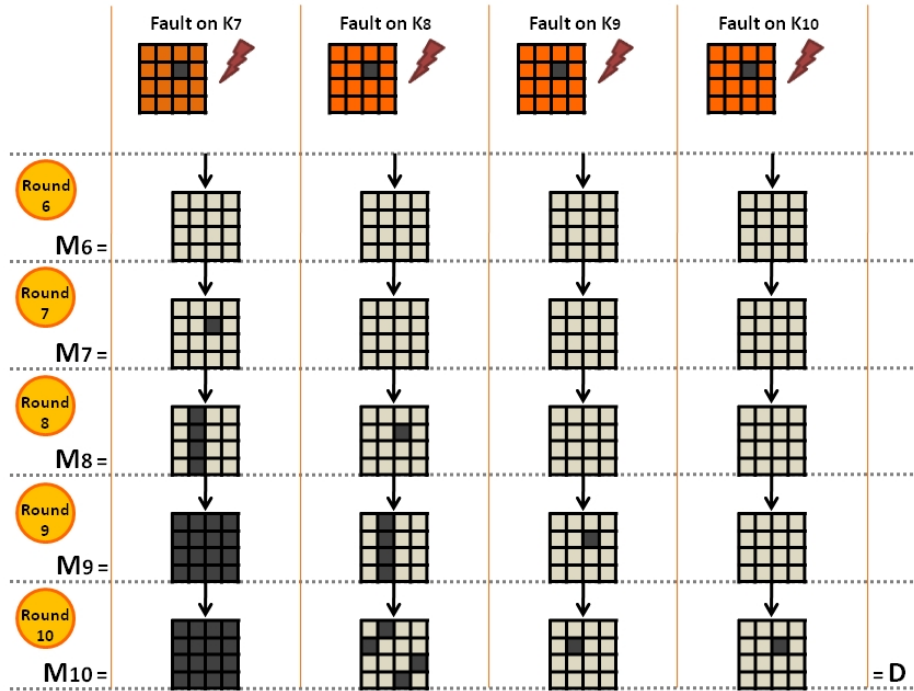


Figure 3.6: Effects of **MixColumns** on fault propagation for one faulty single-bit/byte round key on the temporary ciphertext at the end of each round and at the end of the algorithm.

Consequently, the number of faulty bytes on the final ciphertext may lead the attacker to discover faulty round keys. Table 3.1 shows the relation between observed faulty bytes on the ciphertext and potential faulty round keys.

Table 3.1: Potential faulty K_i s as a function of observed faulty ciphertext bytes.

number of faulty C bytes	potential faulty round keys			
	K_{10}	K_9	K_8	previous round keys
1, 2, 3	✓	✓		
4, ..., 15	✓	✓	✓	
16	✓	✓	✓	✓

In the experiments with memory access, we applied the temporal accuracy to the third step of target exploration. Contrary to the previous experiment, by applying the temporal accuracy on t_8 at the beginning of exploration, we could discard the logical effects of “early faults” occurring in any K_i preceding K_8 . The top part of figure 3.7 shows this exploration.

Then, we searched for spatial accuracy in order to target only the first single-byte on K_8 , as shown at the bottom of figure 3.7.

Finding the SRAM area containing K_8 and properly tuning the laser’s parameters are more time consuming in comparison with the previous experiment. Nevertheless, we could successfully inject required faults for two plaintexts and perform Piret-Quisquater’s DFA blindly.

3.2.4 Conclusion

This attack was usually regarded as one of the most effective fault attacks on AES, as it only requires two faulty ciphertexts. This effectiveness comes at the price of stringent fault injection assumptions.

In summary, this attack can be implemented, even when the laser spot is wider than the SRAM’s cell. When it is physically impossible to target a single-byte because the beam hits a few other bytes, careful spatial and temporal coordination may allow the opponent to deceive the encryption process. By logically considering only a single-byte fault that corresponds to the attack scheme. This assesses the reality of Piret-Quisquater’s scenario on unprotected chips.

These experiments also apply to other attacks (e.g. [Giraud 2005],[Moradi 2006]) and underline the possibility of modifying memory cells in the absence of counter-measures.

Besides, a verification of the injected faults shows that all the single-byte faults of this experiment did not affect more than one bit on each byte. In other words, all the single-byte injected faults in this experiment were also single-bit faults. We already gave the reasons for this in section 2.3. So, in the next section, we explore the possibility of performing a single-bit level DFA on AES.

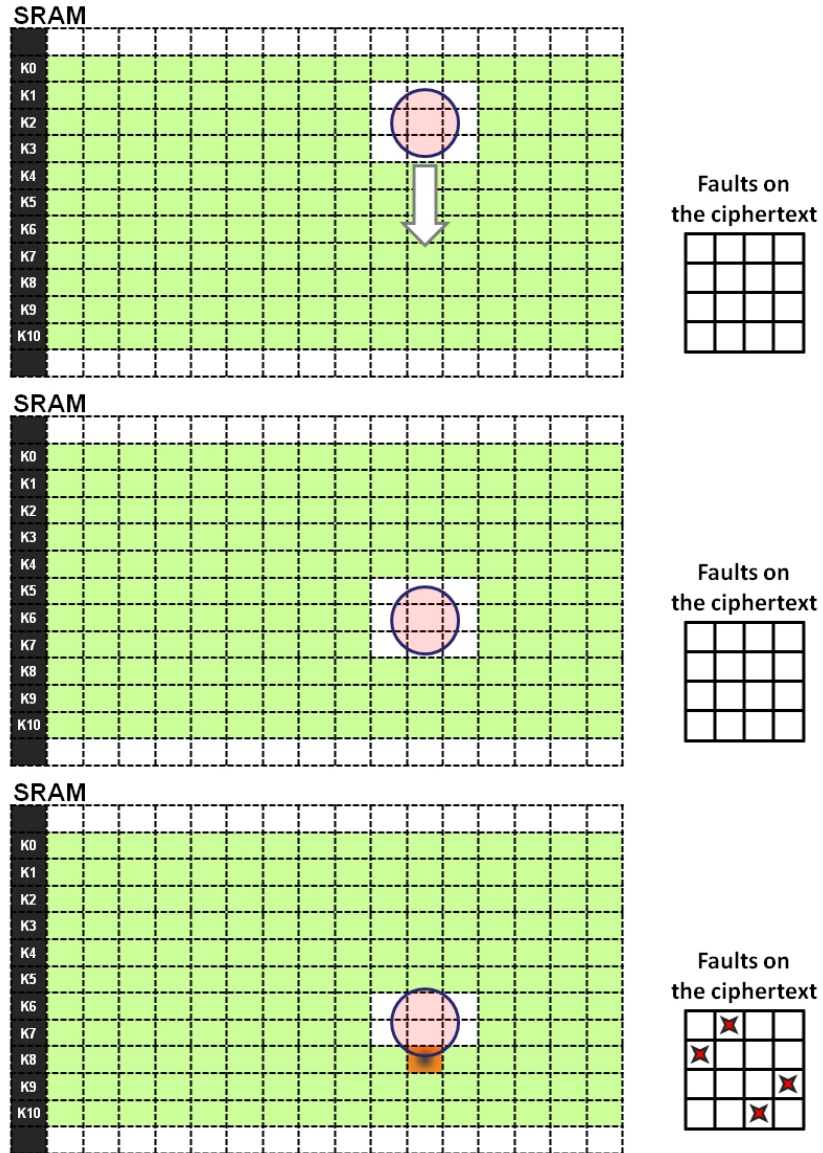


Figure 3.7: Exploration process for Piret-Quisquater's DFA by blind experiment. The laser emission time was set to t_8 during exploration.

3.3 Giraud's Single-Bit Differential Fault Analysis

In the previous section, we demonstrated that a single-byte DFA can be carried out by a surgical laser fault injection even if the laser spot creates several faulty bytes. In this section, we examine the feasibility of performing a single-bit DFA on our embedded AES.

3.3.1 Attack Scheme

Christophe Giraud has described a single-bit and a single-byte DFA on AES in [Giraud 2005]. The single-bit attack requires the injection of a single-bit fault into a specific byte of the temporary ciphertext result of the penultimate round (M_9). Figure 3.8 shows this attack requirement.

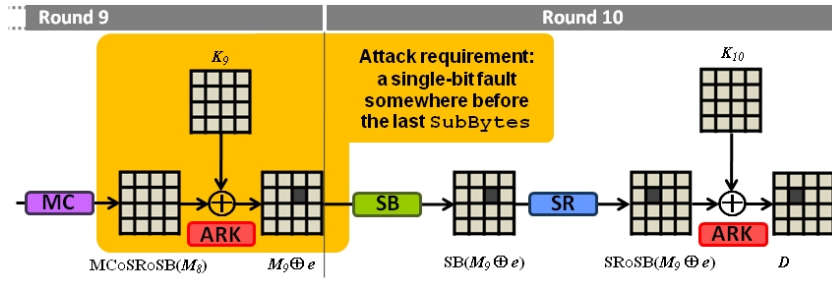


Figure 3.8: Giraud's bit DFA.

To discover one byte of K_{10} , the attack requires the repetition of a single-bit fault, for at least three different plaintexts. The three faulty results are then compared to their corresponding correct ciphertexts in order to infer key information.

During normal processing, the value of each ciphertext (C) byte is calculated by xoring the corresponding K_{10} byte value with the temporary value resulting from the application of **SubBytes** and **ShiftRows** to the M_9 corresponding byte. Equation 3.1 shows this operation:

$$C = SR \circ SB(M_9) \oplus K_{10} \quad (3.1)$$

For the sake of clarity, we consider all subsequent equations *bytewise*, thereby abstracting away **ShiftRows** operations that do not affect individual byte values. Thus, equation 3.1 can be rewritten as 3.2:

$$C = SB(M_9) \oplus K_{10} \quad (3.2)$$

Let e be a single-bit fault injected before **SubBytes** input of the 10th round, the faulty ciphertext (D) can be expressed as equation 3.3:

$$D = \text{SB}(M_9 \oplus e) \oplus K_{10} \quad (3.3)$$

[Giraud 2005] observes that a xor between a faulty and a correct ciphertext reveals a difference ($\Delta = C \oplus D$) corresponding to a set of hypotheses on the corresponding M_9 byte value before the attack, and on the injected single-bit fault e :

$$\Delta = \text{SB}(M_9 \oplus e) \oplus \text{SB}(M_9) \quad (3.4)$$

The equation 3.4 will yield a set of hypotheses on possible M_9 and e value-pairs. Using the equation 3.5, a corresponding K_{10} value can be substituted for each pair of (M_9, e) values.

$$K_{10} = \text{SB}(M_9 \oplus e) \oplus D \quad (3.5)$$

Each pair of a faulty ciphertext and its corresponding correct ciphertext for one byte gives a set of hypotheses for the value its corresponding K_{10} byte. By repeating the fault injection for at least three different plaintexts, the opponent creates the same number of hypothesis sets for the corresponding K_{10} byte value. Then, sets are intersected to spot the single hypothesis that reveals the corresponding K_{10} byte. With a probability of about 97%, three plaintexts suffice to discover a byte of K_{10} [Giraud 2005]. Otherwise, the opponent iterates the process for more plaintexts until the sets' intersection reaches a singleton. Figure 3.9 shows how an intersection between hypothesis sets leads to a unique value for a K_{10} byte.

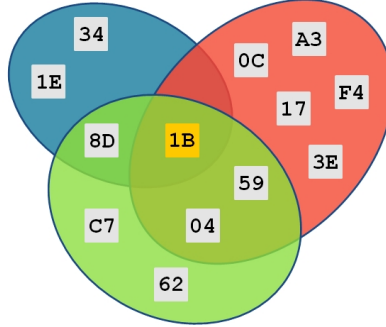


Figure 3.9: An intersection between three hypothesis sets for a K_{10} byte value.

After finishing this operation for one byte of K_{10} , the procedure is repeated to discover K_{10} 's remaining bytes. Finally, $K = K_0$ is inferred by reversing the key schedule.

In a practical experiment, it is possible that a new set of hypotheses does not have any intersection with the previous ones. In this case, one of the sets does not correspond to the attack scheme, *i.e.* one of the injected faults is not single-bit.

However, the opponent may perform a union operation between the sets, instead of intersection operation. The incorrect hypotheses will be easily discarded when the following intersection operation is performed with a new hypothesis set.

3.3.2 Practical Experiment

Now, we survey how we can implement Giraud's single-bit attack on our circuit.

Like the previous attack, here again this fault can be injected via the adequate round key, *i.e.* K_9 . According to figure 3.10, a single-bit fault on K_9 satisfies the attack requirement.

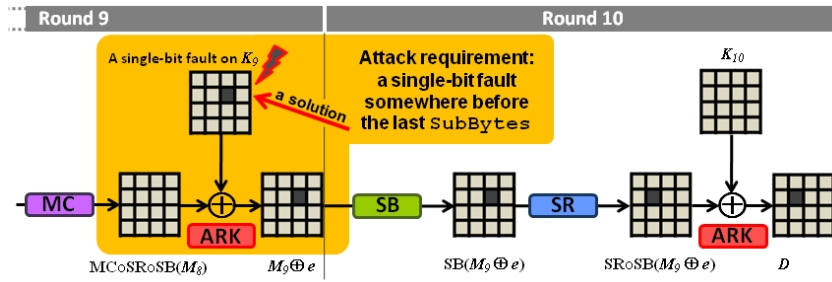


Figure 3.10: Giraud's bit DFA.

For this experiment, it is required to inject a single-bit fault on one K_9 byte at each laser emission. Each faulty byte of K_9 discovers the content of its respective K_{10} byte after at least three experiments.

The **ReadRAM** command, described in section 2.2 helped us to read and detect all the faults injected on the round keys after each laser emission. Despite fine-grained energy and spatial control, we detected faults in keys neighboring K_8 .

To overcome the problem of faulty bytes in keys neighboring K_9 , we again used the three step solution, with a slight change: We performed temporal positioning in the first step. It reduces the number of faulty bytes on the final ciphertext and so it was easier to refine spatial coordinates, for injecting a single-bit fault on K_9 with both options of memory access or blind tests.

Figure 3.11 demonstrates our spatial explorations. As shown at the top of figure 3.11, we searched the target bytes on K_9 . Then, by fine-tuning the spatial beam localization, we tried to restrict the injected faults uniquely on K_9 bytes and protect K_{10} from any faults. The fault injection is performed during the brief time period of t_9 , as shown in figure 3.12. Therefore, during the exploration, any early fault is not logically entered into the encryption.

Our experiments were conducted on our chip's front side, with a $20\times$ lens, a green laser spot of $3.75 \times 3.75 \mu m$ and $\simeq 15pJ$ energy per shot (at laser emitter exit).

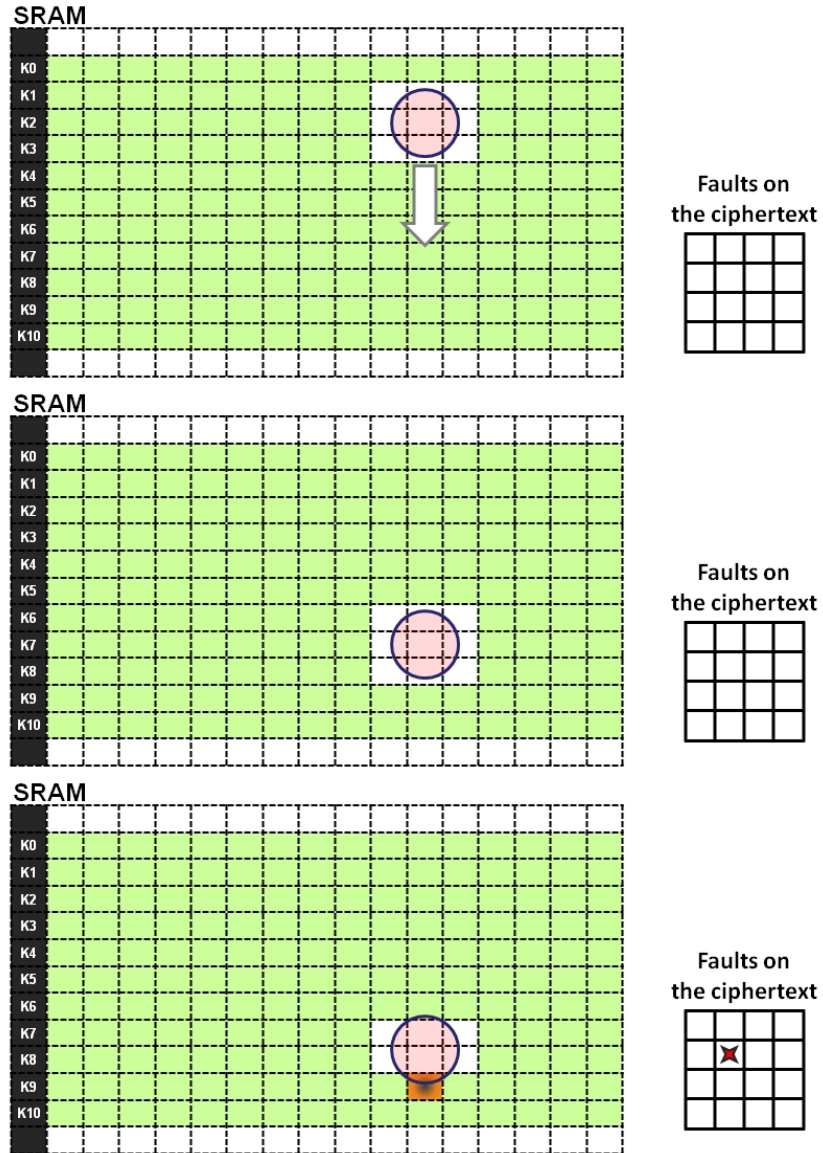


Figure 3.11: Exploration process for Giraud's Single-Bit DFA. The laser emission time is set to t_9 during exploration.

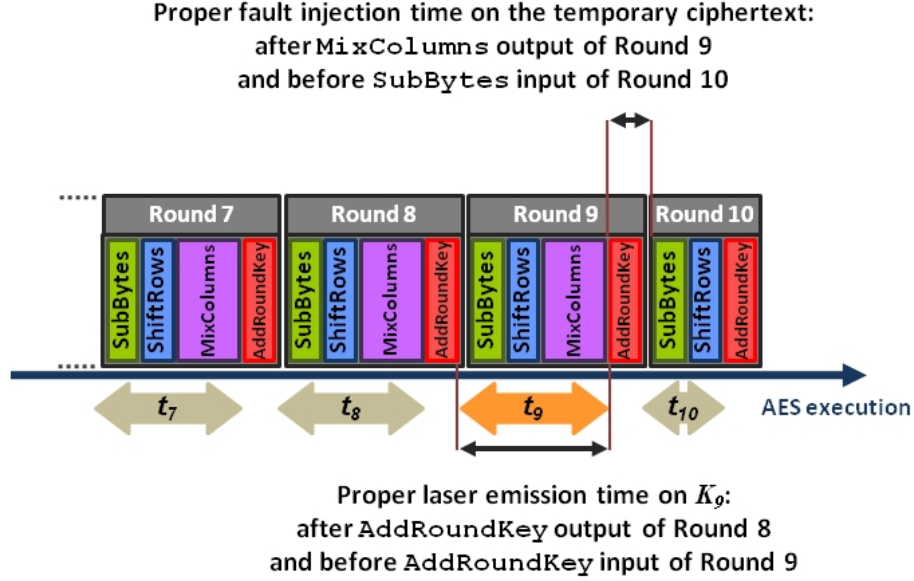


Figure 3.12: Giraud's bit attack timing.

3.3.3 Conclusion

We implemented a single-bit Giraud attack [Giraud 2005] using laser fault injection. While this is not the most effective fault attack on AES, this scenario is usually regarded as the *most difficult*, since it requires limiting the attack to *one single-bit*. This is much more stringent than most other AES fault attacks (e.g. [Piret 2003], [Dusart 2003], [Blömer 2003]) that target an entire byte, regardless of the number of faulty bits.

According to the results of the two previous sections, implementing Piret-Quisquater and Giraud's single-bit DFA, were successful, even when targeting a single-bit or a single-byte is physically impossible (because the beam hits a few other bytes). Careful spatial and temporal coordination may allow the opponent to exclude extra faults and deceive the encryption process, by considering logically only the required fault.

Now, the problem is: If, in anyway, the opponent cannot limit the faults to only one single-bit or single-byte on the targeted round key, is it still possible to perform DFA? In the coming section, we investigate this problem.

3.4 An Extended Single-Bit DFA for Multiple-Byte Faults

In the previous section, we demonstrated a single-bit DFA by laser on AES. We showed that by a surgical laser fault injection and by accurate attack time adjust-

ment, a single-bit fault can be obtained logically between several physical faults. In this section, we describe an extended case, when a laser spot injects single-bit faults on several bytes of targeted round keys. In this case, the opponent cannot reduce the number of faulty bytes. Then, we discuss the possibility of performing bit-level DFA.

3.4.1 1st Scheme for the Extended Attack

The logical implementation of SRAM memory cells is usually represented in a very simple way as a table of consecutive bytes. However, the physical architecture of SRAM memories is much more sophisticated. The reason is the need for optimizing the length and the area of address decoders and data buses.

We performed some preliminary experiments on our circuit's SRAM and reported the results in section 2.3. We discovered the physical bit and byte implementation order during our experiments.

In 3.2.2, laser faults are injected on several bytes, but the content of faulty bytes showed only a one-bit difference with their original (*i.e.* non faulty) value. In fact, we guessed that in several physical implementations of SRAM, the bits of the same value are designed and built close together for a block of bytes in the memory array.

Our fault injection experiments on SRAM between two consecutive SOSSE commands for writing and reading a part of the memory confirmed this guess during the preliminary tests reported in section 2.3.

In the implementation of our microcontroller, the distance of two bit cells of the same value in a block of 256 bytes is much closer than the distance of a bit with its neighboring bits of the same byte. So, in the case of a surgical laser fault injection, the chance of creating single-bit faults simultaneously, on different bytes, is very high. This property helped us to successfully perform Giraud's bit DFA in the subsection 3.3.2.

On the other hand, after `MixColumns` exit of the 9th round, there are no more `MixColumns` steps in the remaining AES encryption operations. So, each byte of the temporary ciphertext stays independent during remaining operations. Consequently, injecting single-bit faults on several bytes of K_9 does not change the attack scheme and even speeds up the DFA.

Therefore, it suffices to protect K_{10} bytes from any fault and to discard the logical effect of faults on previous round keys by temporal accuracy. Figure 3.13 shows a logical representation of SRAM bytes for this extended attack. This attack results in several single-bit faulty bytes on K_9 . If any fault is not entered in the encryption through K_{10} , the opponent can perform Giraud's bit DFA successfully. This is also an advantage for Giraud's Bit DFA that reduces the required pairs of corresponding correct and faulty ciphertexts for parallel faulty bytes.

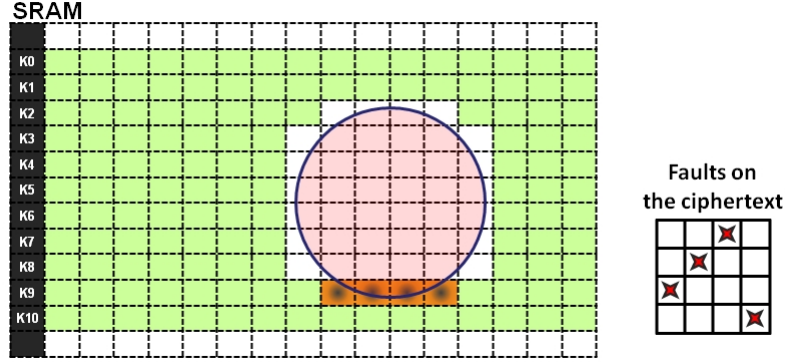


Figure 3.13: Exploration process with a big laser spot. The laser emission time is set to t_9 during exploration.

3.4.2 2nd Scheme for the Extended Attack

Figure 3.13 showed how the opponent can limit fault injection to several bytes of K_9 , protect K_{10} bytes and meanwhile discard faults on other round keys. Now, we assume a more complicated case, where the opponent cannot completely protect K_{10} from fault injection due to the large size of the laser spot. Therefore, few single-bit faults are injected on the K_{10} bytes. We assume again that the injected faults are single-bit.

In this attack, for each faulty K_{10} byte, its corresponding byte on the faulty ciphertext (D) is calculated as equation 3.6:

$$D = \text{SB}(M_9) \oplus (K_{10} \oplus e) \quad (3.6)$$

Consequently, the faulty bytes on ciphertext due to a faulty K_{10} byte have only a single-bit difference with their correct value.

In this context, Δ shows a single-bit difference that corresponds to the injected fault e in K_{10} :

$$\Delta = C \oplus D = e \quad (3.7)$$

According to the equations 3.2, 3.6 and 3.7, for any faulty byte on the ciphertext, in about 97.66% of cases, if Δ shows a single-bit difference between C and D , the faulty key value comes from K_{10} or else it comes from K_9 .

Therefore, by using equation 3.7, the opponent can classify the faulty bytes on the ciphertext to “ K_9 -related” and “ K_{10} -related” fault classes. The class of K_9 -related faults refers to all the faulty bytes with more than one bit differences in comparison to their correct values. Although, the class of K_{10} -related faults contains all the faulty bytes with only one bit difference. As the contents of this class do

not correspond to Giraud's bit fault model, they must be excluded from the list of faulty bytes for the DFA.

3.4.3 Practical Experiment

Figure 3.14 shows a simple representation of this attack. On the top part of figure 3.14 the laser beam hits few bytes on K_{10} and they cannot be discarded by the opponent. So, the opponent can classify faulty bytes on the ciphertext according to their difference (Δ) in comparison to their original values, as shown at the bottom of figure 3.14.

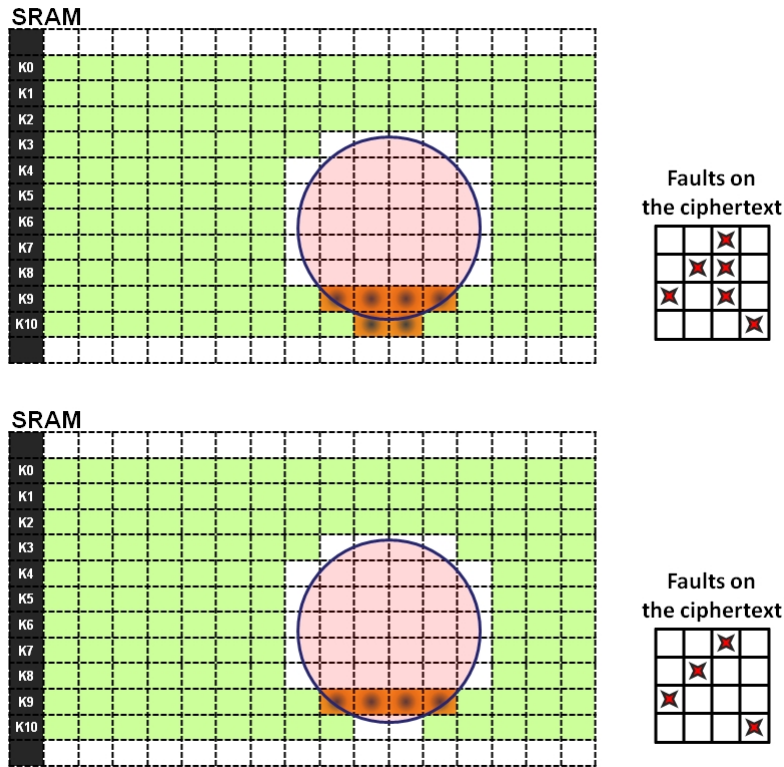


Figure 3.14: Exploration process and result classification. The laser emission time was set to t_9 during exploration. Then, the classification process excluded K_{10} -related faults.

Therefore, the opponent exploits only the class of K_9 -related faults as shown in figure 3.15 to make the assumptions on K_{10} values.

However, two other cases are also possible:

1. A faulty byte on the ciphertext is the result of one faulty byte on K_9 and another one on K_{10} . In this case, equation 3.2 cannot discover the effect of

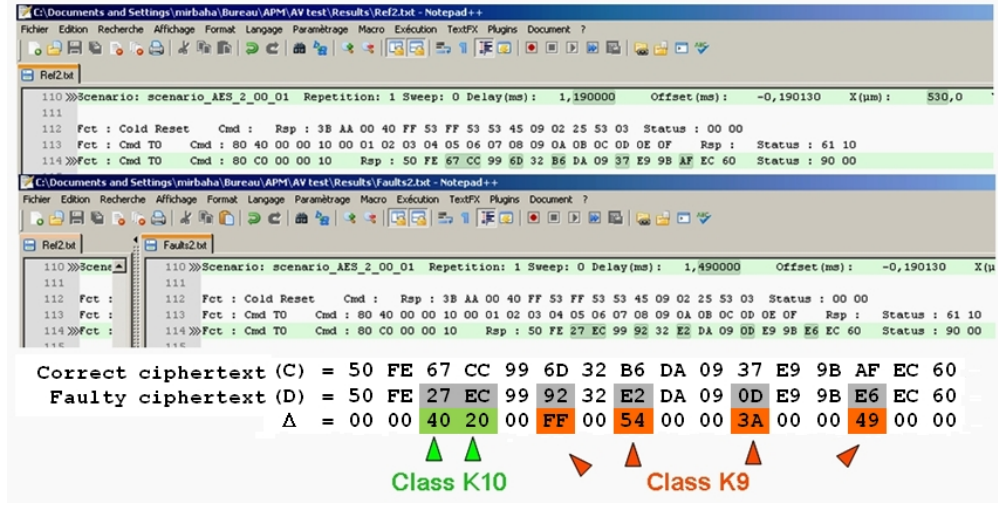


Figure 3.15: Classification of faulty bytes on the ciphertext. Faults are separated into two classes of corresponding (K_9 -related) and non-corresponding (K_{10} -related) faults for Giraud’s bit DFA.

K_{10} and it will be classified as a K_9 -related fault. So, it creates some false assumptions about K_{10} values. Thus, by intersection operations, the false assumptions will be discarded from the set. However, usually an additional pair of a faulty and its corresponding correct ciphertext, is needed to reduce the number of assumptions to a single one. Otherwise, an exhaustive search will be needed for the remaining assumptions in order to examine them and find the correct one.

2. A faulty byte on K_9 creates only a single-bit difference as fault on its corresponding value on the ciphertext. According to our calculations, there are only 48 cases, among 2048 possibilities, for a single-bit fault on a M_9 byte that lead to a single-bit fault on the final ciphertext. In this very exceptional case (representing only 2.34% of possibilities), the faulty byte will be classified by error in the class of K_{10} -related faults. So, an additional pair of corresponding faulty and correct ciphertexts will be needed to reduce the number of assumptions to a single one. Otherwise, like the previous case, an exhaustive search will be needed for the remaining assumptions in order to examine them and find the correct one.

3.4.4 Conclusion

In summary, we could successfully perform Giraud’s DFA using a limited set of faults that correspond to K_9 faults and omit other faults that exist physically on previous and next round keys. These results correspond to Giraud’s bit DFA with an extended possibility to retrieve the key from multiple faulty bytes. We successfully performed

this experiment without reading the memory during the test. We explored only faulty values on the ciphertext in order to classify them and to reveal K_{10} values. Our experiments were conducted on our chip's front side, with a $20\times$ lens, a green laser spot of $4.375 \times 5.625 \mu\text{m}$ and $\simeq 19.7 \text{ pJ}$ energy per shot (at laser emitter exit).

The reproducible single-bit fault injection by big laser spots and Giraud's bit DFA are more feasible on unprotected chips than they are usually considered to be. In addition, separate implementation of bit blocks is a weakness for the security of SRAM contents against single-bit fault injection. So, developing the proper counter-measures against laser fault attacks is necessary for the security of cryptographic circuits.

In the next section, we compare the feasibility of the three previous DFA attacks.

3.5 Feasibility Comparison of Previous Attacks

Of the different DFA methods on the AES, the Piret-Quisquater's technique is usually considered to be one of the most effective attacks. To be exploited, Piret-Quisquater's DFA only requires two sets of faulty ciphertexts and their corresponding correct results. Depending on the location of the targeted byte, at the beginning of the 9^{th} or the 8^{th} round, the two sets of faulty results can reveal respectively, either 4 bytes or all 16 bytes of the key.

The strength of Giraud's bit DFA is classified after Piret-Quisquater's method. Because Giraud's bit DFA requires at least three pairs of corresponding faulty and correct ciphertexts. In addition, the injection of a single faulty bit is usually considered more difficult than the injection of a single faulty byte, and even sometimes infeasible.

However, our results reported in the previous sections showed that the hardware feasibility of such attacks is totally different from what is usually considered. In fact, we demonstrated that on several physical implementations of SRAM memories, the feasibility of injecting several single-bit faults on different bytes is more likely than the targeting of several bits of a same byte. This strength comes from the design of several memory array layout implementations that facilitate the feasibility of single-bit faults on one or on a few bytes (Please refer to figure 3.16, taken from [Pavlov 2008]). The hardware implementation has an important consequence for calculating the feasibility of these attacks.

3.5.1 Review of Piret-Quisquater's Experiment

The Piret-Quisquater's attack requires only one faulty byte at the beginning of the 8^{th} or the 9^{th} rounds. Any more faulty byte in the same area may change the obtained fault and neutralize the DFA process. Some exceptions can be considered on K_9 , however, the feasibility remains questionable when the opponent has not accurate equipment in comparison to the fabrication technology size for injecting faults. Considering K_8 , the model can be extended to cover a maximum of 4 distinct

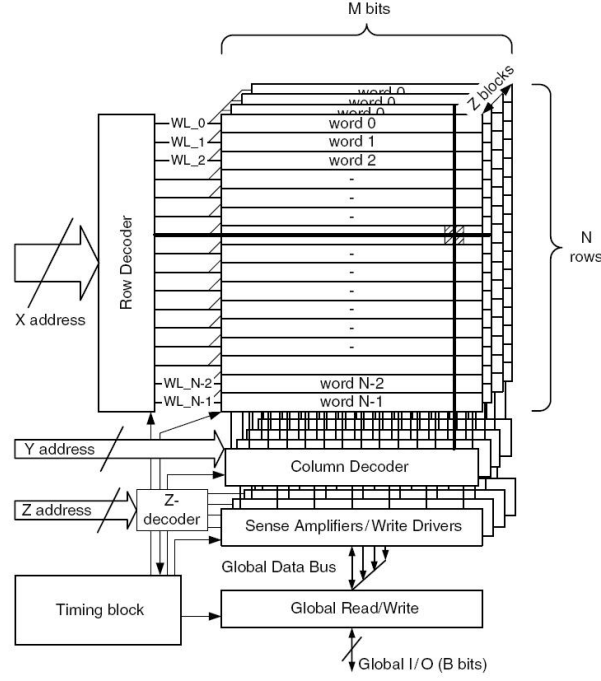


Figure 3.16: A SRAM block diagram.

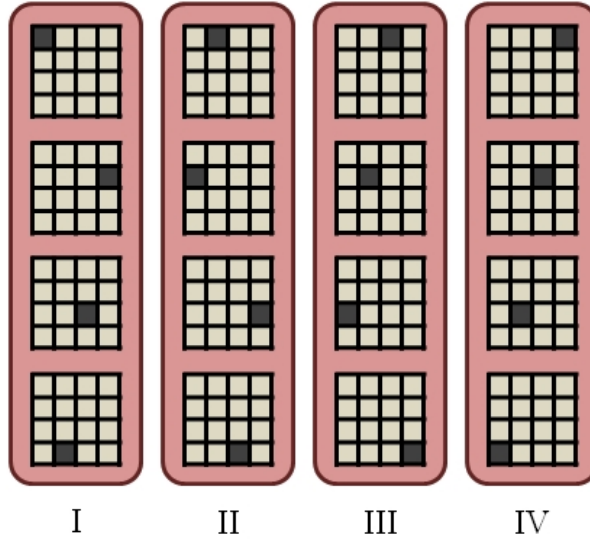


Figure 3.17: Four different sets of possible faulty bytes on K_8 . When more than one faulty byte is injected from an individual set into K_8 , their effects create multiple faults on the respective four bytes after **MixColumns**. Therefore, Piret-Quisquater's DFA process cannot find the key values.

bytes under restricted conditions. Figure 3.17 shows 4 different sets of possible faulty bytes on K_8 . For the cases with more than one faulty byte on K_8 , the model requires that the faulty bytes don't cover other bytes of the same set. Otherwise, the effects of multiple faults from a single set will be repeated on the same bytes of the temporary ciphertext. So, the faults on the final ciphertext can no longer reveal the correct values of the key during DFA processes, as shown on figure 3.18.

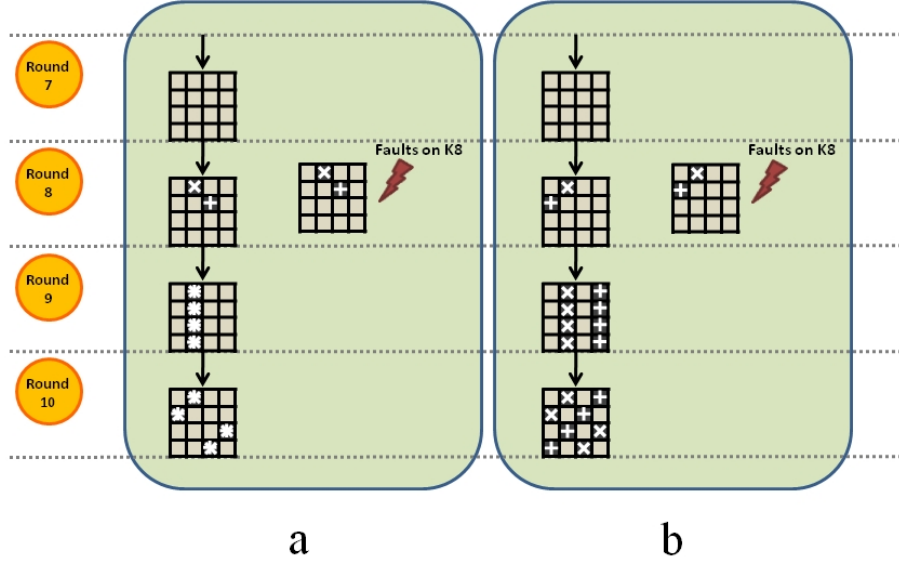


Figure 3.18: Description of additional faulty bytes on K_8 : In part (a), when two faulty bytes are injected from the same set into K_8 , their effects change the content of the same column twice, after the next `MixColumns`. Therefore, Piret-Quisquater's DFA process cannot find the key values. Besides, part (b) shows that when two faulty bytes are injected from two different sets into K_8 , their effects speed up the Piret-Quisquater's DFA process.

So, is it still feasible to perform Piret-Quisquater's DFA when one or several additional faulty bytes appear on the targeted key and/or the next ones? The solutions reported in 3.2 no longer seem to be successful. We consider four different cases:

3.5.1.1 One additional faulty byte on the targeted round key

If a second single-byte fault is injected into K_8 , two different situations may occur:

- The fault is injected into one of three other bytes that enter together into the targeted faulty byte in the next `MixColumns` step. In this case, the additional fault cannot be detected immediately by the opponent, because the number of

faulty bytes on the final ciphertext remains four. Besides, this additional fault eliminates the effect of the targeted faulty byte and disrupts the DFA process.

- The fault is injected into one of 12 other bytes. In this case, the additional fault can be detected immediately by the opponent, because it creates four new faulty bytes on the final ciphertext. So, it can accelerate the DFA process by reducing one of the needed faulty ciphertexts.

3.5.1.2 Two or more additional faulty bytes on the targeted round key

For each of these faults, the previous situations are imaginable. However, it will be very difficult to check that each quadruple of faults is the product of only one faulty byte on the K_8 .

3.5.1.3 One additional faulty byte on the next round key

If a second single-byte fault is injected on K_9 , two other situations may occur:

- The fault is injected into one of the four bytes corresponding to the K_8 faulty byte after the `MixColumns` step. In this case, the additional fault cannot be detected immediately by the opponent, because the number of faulty bytes on the final ciphertext remains four. Besides, this additional fault eliminates the effect of the required (targeted) faulty byte and disrupts the DFA process.
- The fault is injected into one of 12 other bytes. In this case, the additional fault can be detected immediately by the opponent, because it creates only one new faulty byte on the final ciphertext.

3.5.1.4 Two or more additional faulty bytes on the next round key

For each of these faults, the previous situations are imaginable. However, it will be very difficult to verify that the desired quadruple of faulty bytes is the product of only one faulty byte on the K_8 and does not have any effect from K_9 faults.

Therefore, considering K_8 , the DFA model can be extended to cover a maximum of 4 different faulty bytes under restricted conditions. Figure 3.17 shows 4 different sets of possible faulty bytes on K_8 . When more than one faulty byte exists on K_8 , the model requires that only one faulty byte exists in each set of faults. Otherwise, the effects of multiple faults from a single set will be repeated on the same bytes of the temporary ciphertext. So, the faults on the final ciphertext can no longer reveal the correct values of the key during DFA processes, as shown in figure 3.18.

Therefore, additional faulty bytes on the targeted round key or the next one render Piret-Quisquater's DFA unable to exploit the attack. Piret-Quisquater's DFA model is very dependant on its required model. The solutions demonstrated in 3.2.2 cannot eliminate the effects of additional faults on K_8 and K_9 .

3.5.2 Review of Giraud's Bit Experiment

Is it still feasible to perform Giraud's bit DFA when the opponent cannot protect neighboring bytes on the targeted round key from one or a few additional faults?

Similarly, we consider four different cases:

3.5.2.1 One additional faulty byte on the targeted round key

If a second single-bit fault is injected on K_9 , two different situations may happen:

- The fault is injected into the same targeted byte: In this case, the obtained faulty result creates wrong assumptions and cannot lead to the key value. Nevertheless, an additional pair of corresponding faulty and correct ciphertexts can eliminate the effect of the wrong faulty result in the assumption set.

This kind of wrong fault cannot be detected immediately by the opponent, but it is very exceptional in practice. In the case of a very limited fault injection, the chance of two single-bit faults on neighboring bytes is much more probable than multiple-bit faults on the same byte. Because in several memory array layout implementations, the blocks of same-value bits are physically implemented for a set of bytes. Each bloc (of the same-value bits) is separated and kept at a distance from the bits of other values.

- The fault is injected into any of 15 other bytes. In this case, the additional fault can be detected immediately by the opponent, because it creates one new faulty byte on the final ciphertext. So, it can accelerate the DFA process by reducing one of the needed faulty ciphertexts for a second faulty byte.

3.5.2.2 Two or more additional faulty bytes on the targeted round key

For each of these faults, the previous situations are imaginable. However, the feasibility of an additional faulty bit on the principle single-bit faulty byte is very remote. However, any additional faulty byte (containing only a single-bit fault), even up to 15 more bytes, accelerates the DFA process (by reducing the same number of needed faulty ciphertexts).

3.5.2.3 One additional faulty byte on the next round key

If a second fault is injected on the K_{10} , it may very probably be a single-bit fault for the reasons of memory array layout design in our circuit and in several other memory implementations (figure 3.16). So, two different situations may happen:

- The fault is injected on the same K_{10} byte that corresponds to the single-bit fault on K_9 : In this case, the additional fault cannot be detected immediately by the opponent, because the number of faulty bytes on the final ciphertext

remains one. Besides, this additional fault eliminates the effect of the targeted faulty byte and disrupts the DFA process. However, an additional pair of corresponding faulty and correct ciphertexts may discard the effect of the wrong faulty result in the assumption set.

- The fault is injected in one of 15 other bytes: In this case, the additional fault can be detected immediately by the opponent and discarded from the DFA processes. It creates only one single-bit fault on the final ciphertext. So, it can be detected by a comparison between the correct and the faulty ciphertexts.

3.5.2.4 Two or more additional faulty bytes on the next round key

For each of these faults, the previous situations are imaginable. They can be managed by the opponent. There is no limit for single-bit faults on K_9 . Besides, the number of single-bit faults on K_{10} must remain proportionally low.

Therefore, according to these different cases, additional single-bit faults may be managed by the opponent in Giraud's bit DFA.

3.5.3 Conclusion

We compared the feasibility of the Giraud's bit and the Piret-Quisquater's DFA when additional injected faults on the targeted key or the next ones cannot be avoided. The current section is concerned with the practical flexibility of these DFA methods by single-bit/byte laser faults on the stored AES round keys in a SRAM memory. We showed that a laser spot that hits a few more bits than the reference model, may still be effective in carrying out the DFA. After finding the flexibility of Giraud's bit DFA (for additional faults), we presented our extended DFA method, based on spatial and temporal precisions, followed by a classification of the results. Besides, the probability of multiple single-bit faults on different bytes at the penultimate round key in a Giraud's-bit attack is higher than the probability of a multiple-bit fault on the target byte. In this case, all the faulty bit/bytes will stay independent from other ones during the remaining part of the encryption. These multiple faults will increase the speed of analysis and do not disturb the results.

On the other hand, Piret-Quisquater's DFA has a practical weakness that is illustrated clearly in figure 3.18. In an experiment, the opponent performs a single-byte fault attack on K_8 and he obtains the final ciphertext with the corresponding number of faults, *i.e.* 4 faulty bytes. Then, how can the opponent be sure that only one faulty byte is injected into K_8 , and not into an additional byte that has the same effects? On the Giraud's bit model, this problem does not exist; hence, this is a practical advantage in comparison to Piret-Quisquater's DFA. The only weakness of Giraud's-bit attack, is the probability of injecting faults on neighboring round keys. In this case, by using our method for temporal accuracy, all the physical faults on the previous round keys can be discarded from the encryption process. In addition,

our extended attack, described in section 3.4, shows a solution for any injected fault on the next round key (K_{10}).

Our comparison is summarized in Table 3.5.3. Our extended attack, based on Giraud's-bit model, is more flexible, in comparison to the two previous attacks by laser fault injection into SRAM contents. When a few additional faulty bytes are created on the targeted round key and on the following key, our extended Giraud's bit DFA can exploit the faulty results under restricted conditions. Our main conclusions are:

- When the laser beam encounters several bytes, spatial and temporal accuracy may discard the effects of injected faults on previous round keys. In this case, several faulty bytes on the targeted key may increase the performance of Giraud's bit DFA, but disrupt Piret-Quisquater's method.
- When a few faults are injected into the next round keys, a classification between K_9 -related and K_{10} -related faults may exclude the second class faults from Giraud's bit DFA, but a similar possibility does not exist for Piret-Quisquater's DFA and additional faults may neutralize the process.
- The feasibility of Giraud's bit DFA is much higher than what is usually thought for unprotected chips. Moreover, the design of physical layout may particularly improve the possibility of required faults. So, developing the proper countermeasures against laser fault attacks is necessary for the security of cryptographic circuits.

A feasibility study of performing such faults by different categories of opponents, according to their level of technical knowledge and financial budget for equipment, is necessary for developing proper countermeasures.

In the next section, we investigate the feasibility of a more generalized DFA method.

Fault model	Piret-Quisquater's DFA	Giraud's-bit DFA	Our Extended-Giraud's-bit DFA
Target	Single-byte Temporary ciphertext between MixColumns exit of R_8 and SubBytes entry of R_9 K_8	Single-bit Temporary ciphertext between MixColumns exit of R_9 and SubBytes entry of R_{10} K_9	One or several single-bit Temporary ciphertext between MixColumns exit of R_9 and SubBytes entry of R_{10} K_9
Alternative target on the round keys			
Additional fault on the previous round keys	Disrupts the DFA process	Disrupts the DFA process	Discarded by spatial and temporal precisions
One additional fault on the targeted K	80%: Detectable - 20%: Not	Managed	Included in the model
More than one additional fault on the targeted K	Difficult to detect	Managed	Included in the model
One additional fault on the next K	75%: Detectable - 25%: Not	Not managed	Included in the model
More than one additional fault on the next K	Difficult to detect	Not managed	Included in the model

3.6 An Extended Multiple-Byte DFA

We presented our extended Giraud's bit DFA in section 3.3. This attack required single-bit faults on K_9 . When the laser beam is larger than SRAM cells, additional faults on the previous round keys were logically excluded from the encryption by temporal tuning of laser emission. Moreover, additional single-bit faults injected into several bytes of K_9 , did not attenuate the analysis and even sped it up. Furthermore, a few additional faults on K_{10} could be identified and excluded from the DFA process by a simple classification. Nevertheless, when the opponent cannot protect K_{10} from numerous faults on it, our DFA will no longer be functional. So, is there any other solution when the laser beam is too large and encounters several bytes on K_9 and K_{10} ?

3.6.1 Attack Scheme

For a more extended attack on the AES, according to our previous experiments, the first idea may be to shift an additional round and inject the faults at t_{10} as shown in figure 3.19.

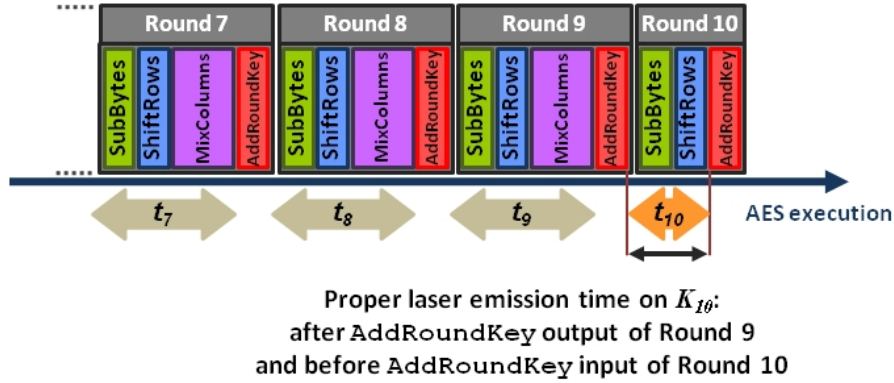


Figure 3.19: Proper attack timing on the K_{10} for excluding the logical effect of eventual faults on the previous round keys.

However, this attack cannot reveal any knowledge of the key by DFA when our fault model is bit-flip or random. In this case, each faulty ciphertext byte can be represented by equation 3.8:

$$D = \text{SR} \circ \text{SB}(M_9) \oplus (K_{10} \oplus e) \quad (3.8)$$

An xor operation between equation 3.8 and the correct ciphertext (equation 3.1) gives only the fault on K_{10} , as shown in equation 3.9. It cannot provide any information about the original key value:

$$\Delta = C \oplus D = e \quad (3.9)$$

Therefore, when the fault model is bit-flip or random (*i.e.* in our case), any fault attack on the round keys must be done on the penultimate round key or before it, in order to be exploitable by DFA.

So, the opponent needs a DFA that exploits faults on K_9 in presence of several faults on K_{10} . In this case, faulty bytes on the ciphertext are sometimes products of two faults on corresponding K_9 and K_{10} bytes. Each pair of corresponding faulty and correct bytes on the ciphertext can be expressed by the equations 3.10 and 3.11:

$$C = \text{SR} \circ \text{SB}(M_9) \oplus K_{10} = \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M_8) \oplus K_9] \oplus K_{10} \quad (3.10)$$

$$D = \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M_8) \oplus K'_9] \oplus K'_{10} \quad (3.11)$$

In this attack, K'_9 and K'_{10} are unknown values corresponding to faulty round keys. We assume that E_9 is the fault on K_9 that has changed it to K'_9 , as shown in equation 3.12. E_9 can be a single-bit or a multiple-bit fault, contrary to e in our previous experiment which was strictly a single-bit fault.

$$K'_9 = K_9 \oplus E_9 \quad (3.12)$$

Therefore, we replace K'_9 values in equations 3.11 according to equation 3.12 and we get a new equation 3.13:

$$D = \text{SR} \circ \text{SB}[(M_9) \oplus E_9] \oplus K'_{10} \quad (3.13)$$

For exploiting successfully this DFA, we need at least three pairs of corresponding correct and faulty ciphertexts. We mark them with a , b and c suffixes: For instance, C^a and D^a for the first pair and M_9^a for their corresponding temporary ciphertext at end of the 9th round.

We perform xor operations between the first faulty ciphertext and each of the following ones and obtain equations 3.14:

$$D^a \oplus D^b = \text{SR} \circ \text{SB}[(M_9^a) \oplus E_9] \oplus \text{SR} \circ \text{SB}[(M_9^b) \oplus E_9] \quad (3.14a)$$

$$D^a \oplus D^c = \text{SR} \circ \text{SB}[(M_9^a) \oplus E_9] \oplus \text{SR} \circ \text{SB}[(M_9^c) \oplus E_9] \quad (3.14b)$$

Then, we factorize **ShiftRows** operations and transfer them to the other side of equations 3.14. Thus, we obtain new equations 3.15:

$$\text{SR}^{-1}(D^a \oplus D^b) = \text{SB}[(M_9^a) \oplus E_9] \oplus \text{SB}[(M_9^b) \oplus E_9] \quad (3.15a)$$

$$\text{SR}^{-1}(D^a \oplus D^c) = \text{SB}[(M_9^a) \oplus E_9] \oplus \text{SB}[(M_9^c) \oplus E_9] \quad (3.15b)$$

Furthermore, we perform xor operations between the first correct ciphertext and each of the other ones in equations 3.10 and thereby, we obtain equations 3.16:

$$C^a \oplus C^b = \text{SR} \circ \text{SB}(M_9^a) \oplus \text{SR} \circ \text{SB}(M_9^b) \quad (3.16a)$$

$$C^a \oplus C^c = \text{SR} \circ \text{SB}(M_9^a) \oplus \text{SR} \circ \text{SB}(M_9^c) \quad (3.16b)$$

We factorize **ShiftRows** operations and transfer them to the other side of the equations 3.16 and we get the new equations 3.17:

$$\text{SR}^{-1}(C^a \oplus C^b) = \text{SB}(M_9^a) \oplus \text{SB}(M_9^b) \quad (3.17a)$$

$$\text{SR}^{-1}(C^a \oplus C^c) = \text{SB}(M_9^a) \oplus \text{SB}(M_9^c) \quad (3.17b)$$

Besides, by using the equations 3.15, we write M_9^b and M_9^c values as functions of M_9^a , K_9 , K_{10} and the correct ciphertexts. So, we obtain the equations 3.18:

$$M_9^b = \text{SB}^{-1}[\text{SR}^{-1}(C^a \oplus C^b) \oplus \text{SB}(M_9^a)] \quad (3.18a)$$

$$M_9^c = \text{SB}^{-1}[\text{SR}^{-1}(C^a \oplus C^c) \oplus \text{SB}(M_9^a)] \quad (3.18b)$$

We replace obtained values for M_9^b and M_9^c from equations 3.18 in equations 3.14. They result in new equations 3.19:

$$\begin{aligned} \text{SR}^{-1}(D^a \oplus D^b) &= \text{SB}[(M_9^a) \oplus E_9] \oplus \\ \text{SB}\{\text{SB}^{-1}[\text{SR}^{-1}(C^a \oplus C^b) \oplus \text{SB}(M_9^a)] \oplus E_9\} \end{aligned} \quad (3.19a)$$

$$\begin{aligned} \text{SR}^{-1}(D^a \oplus D^c) &= \text{SB}[(M_9^a) \oplus E_9] \oplus \\ \text{SB}\{\text{SB}^{-1}[\text{SR}^{-1}(C^a \oplus C^c) \oplus \text{SB}(M_9^a)] \oplus E_9\} \end{aligned} \quad (3.19b)$$

C^a , C^b , C^c , D^a , D^b and D^c have known values.

Finally, we perform an exhaustive search among all of the 2^8 possible values for each M_9^a byte and also among all of the 2^8 possible values for each corresponding E_9 byte. This exhaustive search leads often to a unique value for each M_9^a byte and another unique value for the corresponding E_9 byte. Then, by using the obtained M_9^a byte values and using the equation 3.20, we find K_{10} byte values. The equation 3.20 is calculated from the first correct ciphertext equation 3.10.

$$K_{10} = \text{SR} \circ \text{SB}(M_9^a) \oplus C^a \quad (3.20)$$

Now, we examine the practical experiment for this DFA.

3.6.2 Practical Experiment

For this experiment, the following three steps are required:

1. **Correct encryption:** It is necessary to obtain the correct ciphertexts for at least 3 plaintexts.
2. **Fault injection on K_9 :** The opponent must inject faults on K_9 bytes. In this attack, we cannot exclude the logical effect of any fault on previous round keys. Therefore, it is necessary to physically protect the previous round keys from fault injection, but there is not any restriction on K_9 and K_{10} . The laser emission can be done at any time in t_9 or before. Figure 3.20 shows this attack exploration.
3. **Faulty encryption:** It is necessary to encrypt again the same plaintexts and obtain their faulty ciphertexts without circuit reset.

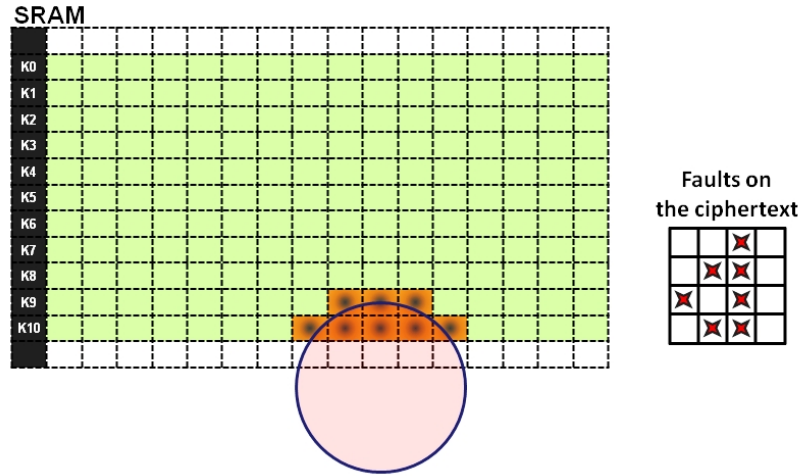


Figure 3.20: Attack exploration for this extended multi-byte DFA. The laser emission time is set to t_9 during exploration.

Therefore, each faulty byte of K_9 reveals the content of its respective K_{10} byte after the calculation described in the attack scheme and using the equations 3.19 and 3.20.

In Piret-Quisquater and Giraud's bit DFA, for each byte and for each plaintext a new fault injection is necessary. Moreover, it is not necessary to repeat the same fault for second and following texts. Although, in this attack, *i.e.* our extended multi-byte DFA, the same faults must be repeated for all the corresponding bytes on all three plaintexts. Consequently, the opponent must avoid a circuit reset after fault injection.

3.6.3 A More Sophisticated Practical Experiment

For our extended multi-byte DFA, we consider a more sophisticated experiment where the opponent cannot physically protect from fault injection all the round keys prior to K_9 . Nevertheless, this attack is still feasible but requires longer practical experiments.

In this case, these different stages must be performed:

1. **Correct encryption:** It is necessary to obtain the correct ciphertexts for at least 3 plaintexts; e.g. C^a , C^b and C^c for M^a , M^b and M^c respectively.
2. **Injecting faults into K_9 and creating the list of faulty ciphertexts:** The opponent launch a series of tests by fault injection on K_9 bytes and must perform two encryptions for each laser emission. The opponent does not need to protect K_{10} from faults.

For the first encryption, it is necessary to perform the laser emission at t_9 in order to exclude the logical effect of any fault on previous round keys. Then, the second encryption is done without circuit reset. Therefore, the faults on the first faulty ciphertext are due to K_9 and K_{10} faults. The second faulty ciphertext includes all the faults on the round keys.

For the first encryptions, the opponent repeats the sequence of 3 plaintexts; i.e. M^a for the first test, M^b for the second one, M^c for the third one. Then once again, he repeats the same sequence: M^a for the fourth test, etc.

For the second encryption, the opponent repeatedly uses the first plaintext; i.e. M^a , as the reference text. Therefore, if at any time, between two previous tests, the second faulty ciphertexts are equal, it means that all the faults on the round keys are similar. Consequently, the first encryptions of these two tests are produced by the same faults on K_{10} and K_9 . The second encryptions are also produced by the same faults on K_{10} , K_9 and prior round keys.

Thus, the opponent creates a list of three values: an index of the first plaintext, a first faulty ciphertext and a second faulty ciphertext. The opponent launches a series of tests and fills the list according to the experiments. As soon as he gets 3 identical second faulty ciphertext values for the tests done with 3 distinguished plaintexts at first encryptions, the experiment ceases.

Now, the opponent uses the four corresponding values for the first encryptions. Each faulty byte of K_9 reveals the content of its respective K_{10} byte after calculation described in the attack scheme and using the equations 3.19 and 3.20.

Figure 3.21 shows this attack exploration. The top part of figure 3.21 refers to the first encryption, when the logical effect of faults on round keys prior to K_9 is discarded. The bottom part of figure 3.21 shows the second encryption, when all the physical faults on round keys are involved in the faulty ciphertext.

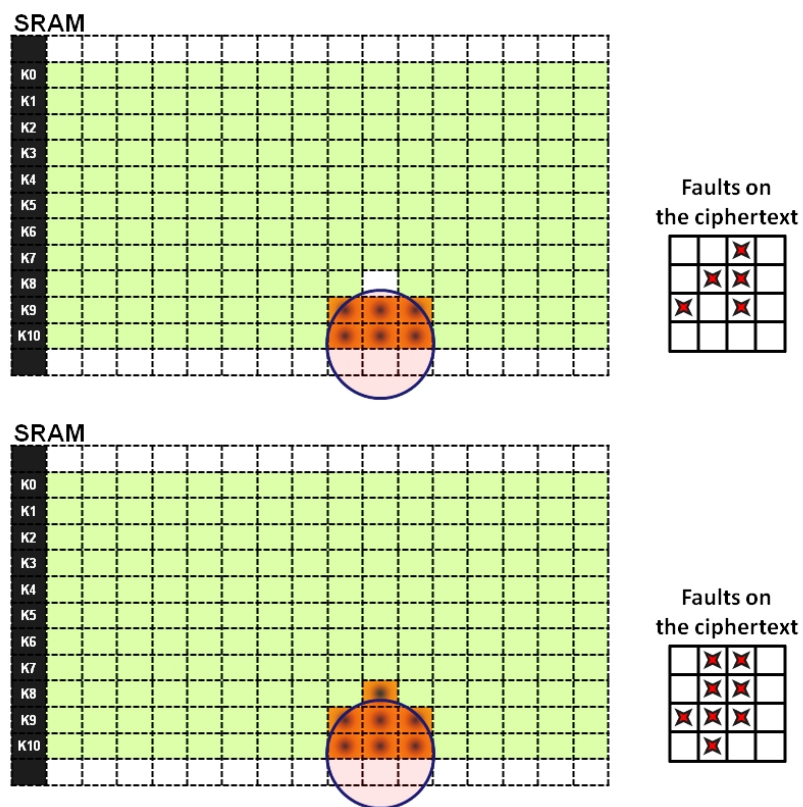


Figure 3.21: The attack exploration for the more sophisticated case of our extended multi-byte DFA. The laser emission time is set to t_9 for the first encryption during exploration. The first and the second encryptions are shown in the top and the bottom parts, respectively.

At the first look, this experiment seems to be very time consuming. However, in practice, for our circuit, the experiment is faster than expected. As the laser target is not moved during experiments and affected bits on the targeted byte are limited to one or two bits, there is a high chance of reproducing similar faults 3 times in a short period.

For instance, we assume that the laser beam targets 7 bytes on the round keys. We also assume that the laser beam often injects between 3 and 5 faults on only two bits of targeted bytes during each emission. In this example, 1512 combinations may occur according to equation 3.21:

$$\begin{aligned} & \text{number of total combinations} = \\ & \sum_{\text{faulty bytes}} \text{number of possible faulty values}^{\text{number of bytes}} \times \binom{\text{total bytes}}{\text{faulty bytes}} = \quad (3.21) \\ & 2^3 \times \binom{7}{3} + 2^4 \times \binom{7}{4} + 2^5 \times \binom{7}{5} \end{aligned}$$

Therefore, if each combination has a chance equal to the other ones, in the worse case, after $1512 \times 2 + 1 = 3025$ tests, some K_{10} bytes can be revealed. For our circuit, this number of experiments can be done in few hours. It is not an ideal attack, but it shows that the single-bit/byte fault model attacks are sometimes possible thanks to tricks, in the absence of proper countermeasures.

3.6.4 Conclusion

In this subsection, we presented our very-extended multi-byte DFA. We showed that even with a very large laser spot that hits many key bytes, it is still possible to reveal the key by DFA.

3.7 Conclusion

In this chapter, we surveyed the feasibility of Piret-Quisquater and Giraud's bit methods by a laser beam that hits several bytes. Our experiments for these single-byte and single-bit methods were successful.

During laser fault injection on the targeted round keys; *i.e.* K_8 for Piret-Quisquater and K_9 for Giraud's bit DFA, we obtained extra faulty bit/bytes on the targeted round key and also on the previous and the following ones, compared to the required model.

By using a laser emission time tuning, we logically excluded the effect of any fault which occurred on the previous round keys. We also protected the following round keys and other bytes of the targeted key by accurate displacements of the

circuit below the laser emitter using an X-Y positioning table with $0.1\ \mu\text{m}$ precision for displacements.

Until finishing this part of our experiment, our main conclusion was that by using a big laser spot that hits several bytes, it is still possible to perform single-byte or even single-bit DFA. Instead of investing in an accurate laser facility, an opponent may use a cheap laser emitter and compensate for its lack of precision by investing in an accurate X-Y positioning table.

Afterwards, we presented two extended DFA models. We showed that even if it is not possible to restrict the fault injection to a single-bit or a single-byte, an intelligent combination between logical and mathematical methods, may facilitate the successful performance of a DFA on AES.

The laser spot diameter and fabrication technology size are important parameters in the success of single-bit/byte fault injection. Although, for all the reasons described in the current chapter, single-bit or single-byte DFA methods are sometimes feasible by using tricks on unprotected chips. They might be realized, even by large laser spots. Therefore, developing the proper countermeasures against laser fault attacks is necessary for the security of cryptographic circuits.

Round Modification Attacks

Contents

4.1 Round Reduction Attacks	77
4.1.1 Previous Studies	77
4.2 Feasible Attack Models on our AES	79
4.2.1 Single Attack Scenarios	80
4.2.2 Cryptanalysis of the Main Attacks	87
4.2.3 Secondary Attack Scenarios	110
4.2.4 Cryptanalysis of Secondary Attacks	117
4.3 Conclusions	117

4.1 Round Reduction Attacks

The Round Reduction Attacks are a category of attacks by algorithm modification using faults. They were noticed by Hamid Choukri and Michael Tunstall at FDTC 2005 [Choukri 2005]. Their work shows a round reduction attack using faults on an AES. Not much research has followed this kind of attacks. The most notable studies are the work of Yannick Monnet et al. presented at FDTC 2006 [Monnet 2006] and the attack of JeanHoon Park et al. reported in June 2011 in [Park 2011].

In this chapter, we demonstrate the possibility of setting up this category of attacks by using laser fault injection. We conducted our experiments on our circuit running SOSSE operating system with an embedded AES.

4.1.1 Previous Studies

4.1.1.1 Attack of H. Choukri and M. Tunstall

According to the experiments of H. Choukri and M. Tunstall [Choukri 2005], if the opponent changes the round counter (therefore RC) of AES to its final value (0 in the [Choukri 2005] case) at the end of the initial round, the ciphertext will be the product of a single round (after the initial round).

According our AES implementation, one of the two different algorithm executions may occur:

1. The ciphertext is the product of the initial round (R_0) and the execution of a middle or normal round (identical to $R_1..R_9$) of AES:

$$D = \text{MC} \circ \text{SR} \circ \text{SB}(M \oplus K) \oplus K_m$$

K_m is the corresponding subkey for the round “m”.

2. The ciphertext is the product of the initial round (R_0) followed by the final round (R_{10}) execution:

$$D = \text{SR} \circ \text{SB}(M \oplus K) \oplus K_{10}$$

In both cases, using two pairs of corresponding faulty and correct ciphertexts, the analysis reveals the secret key. This attack was done with a naive implementation of AES with no countermeasure, embedded on a PIC16F877 microcontroller. The fault injection method was a current glitch on the V_{cc} , at the end of the AES initial round.

4.1.1.2 Attack of Y. Monnet et al.

Yannick Monnet et al. reported in [Monnet 2006] another round reduction attack on two asynchronous cryptoprocessors running the DES algorithm. The round counters used in both of the cryptoprocessors have a multi-rail design. For each of the 16 rounds, a separated signal line is implemented. A 17th signal line also exists to distinguish the end of the algorithm execution. An alarm protects the signal lines and it goes off if more than one signal line is activated.

This attack was done by laser fault injection. Between the two DES asynchronous cryptoprocessors, the model with countermeasures was more resistant to attacks during the experiments. However, this attack is done successfully on both the circuits.

4.1.1.3 Attack of J. H. Park et al.

The attack of JeaHoon Park et al. reported in [Park 2011] is a laser fault attack on an embedded AES using an ATmega128 microcontroller. The AES implementation is based on the proposed one in the NIST reference [NIST 2001].

They reported a successful attack for jumping from R_1 to R_{10} . The faulty execution includes $R_0 - R_1 - R_{10}$. Therefore, it executes an additional round in comparison to the attack of H. Choukri et M. Tunstall [Choukri 2005] that included only $R_0 - R_{10}$.

4.2 Feasible Attack Models on our AES

Considering the implementation of our AES running under SOSSE, embedded on our microcontroller, several possibilities for fault injection are conceivable. For this reason, we refer to the fault models obtained in our previous experiments.

For the feasibility study of these attacks, we review the details of our AES algorithm shown in figure 4.1:

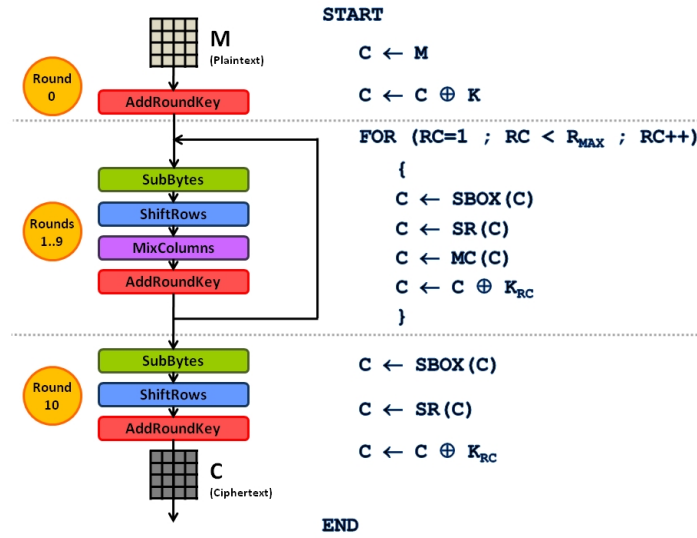


Figure 4.1: Implementation of our AES algorithm.

In our implementation, the round counter is used for counting only the middle rounds (R_m); in other words, the rounds between R_1 and R_9 . The initial round (R_0) and the final round (R_{10}) are implemented separately as shown in figure 4.1. Therefore, even with complete removal of the middle rounds, the initial and the final rounds will be executed.

However, the index of the round key used for **AddRoundKey** operation at any round, even at the initial and at the final round, is same as the round counter value. When the value of RC is between 1 and 10 ($1 \leq RC \leq 10$), the algorithm performs an xor operation between the temporary ciphertext and K_{RC} . But if RC takes a value greater than 10, the algorithm searches the 16 stored bytes in memory that correspond to an address calculated by the same formula for K_{RC} . So, the value of this bloc of 16 bytes will be used, despite the fact that it does not match any valid key value. We recall that any round key with an index greater than 10 cannot exist logically in AES-128. Therefore, the temporary ciphertext is xored with a bloc of unknown values.

According to the laser fault injection model that we implemented successfully during previous experiments, our injected faults may be limited to one or very few

bytes, and usually under a single bit fault model. Therefore, we limit our fault model to this set of hexadecimal values: $\{01, 02, 04, 08, 10, 20, 40, 80\}$. Our study can be extended to the multi-bit faults.

The fault injection moment has distinct effects according to its corresponding algorithm execution moment. For the characterization of these effects, we divide each round execution into distinct moments:

1. **FOR** : At the beginning of the middle rounds loop, the RC value is assigned or increased incrementally. Then, it is compared to its upper limit, the so called " R_{max} " value.
2. **LOOP** : Three transformations of the AES are executed for the current round, before the **AddRoundKey**.
3. **ARK** : The **AddRoundKey** transformation is executed for the current round.
4. **COMEBACK** : When the **AddRoundKey** is finished and the program returns to the loop's beginning.

We show these various moments of our AES algorithm execution in figure 4.2.

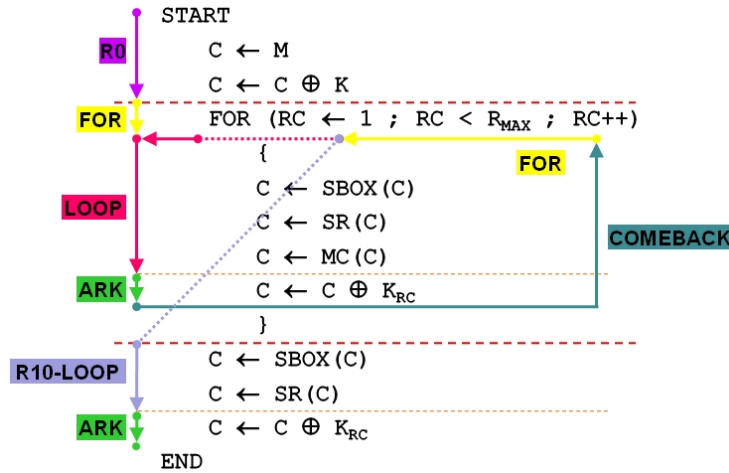


Figure 4.2: Various moments of our AES algorithm execution.

4.2.1 Single Attack Scenarios

Two scenarios are conceivable for changing the number of total rounds in our AES algorithm implementation by fault injection. The targets are the round counter (RC) value and the total round number reference (R_{max}).

In this subsection, we describe the so called *single attack scenarios*, because they require only one fault injection for each plaintext.

4.2.1.1 Attacks on The Round Counter Value

This attack scenario changes the round counter during AES execution. Therefore, it changes the index of the current executing round. Depending on the moment of fault injection, various changes can occur in the algorithm execution. Changing the RC value often leads to altering the total number of executed rounds, by adding, by suppressing or by even executing repetitively several rounds:

- If $RC \oplus e < RC \Rightarrow$ *Round addition or repetitive execution of several rounds*. For instance: if $RC = 7$ and $e = 2$ then the new $RC = 5$ and the AES execution will be:

$$R_0..R_5-R_6-R_5-R_6-R_7..R_{10}$$

The rounds 5 and 6 will be executed twice and the total number of executed rounds will be increased to 12.

- If $RC \oplus e > RC$ and $RC < R_{max} - 1 \Rightarrow$ *Round reduction*. For example: if $RC = 4$ and $e = 2$ then the new $RC = 6$ and the faulty AES execution will be:

$$R_0..R_3-R_6..R_{10}$$

Therefore the rounds 4 and 5 will be skipped and the total executed number of rounds will be reduced to 8.

- If $RC \oplus e > RC$ and $RC = R_{max} - 1 \Rightarrow$ *Round alteration ; i.e. the faulty RC value does not change the total number of rounds, but it alters the **AddRoundKey** transformation of the final round. If the fault is injected during LOOP time of $R_{max} - 1$, it also alters the **AddRoundKey** transformation of the penultimate round. For instance: if $RC = 9$ and $e = 2$ is injected during LOOP time of R_9 , then the new $RC = 11$ and the AES execution will be:*

$$R_0..R_8-R_{m=11}-R_{f=12}$$

So, the total number of executed rounds will remain 10, but the penultimate round and the final round will be xored by invalid subkey values, such as K_{11} and K_{12} during **AddRoundKey**.

We summarize these effects according to different stages of algorithm execution and show them in table 4.1.

Table 4.1: Effects of fault injection on the round counter according to the different stages of the AES algorithm execution

Stage	Effects
FOR	- Change of the RC value - Addition or reduction of total rounds (even repetitive execution or suppression of several rounds)
LOOP	- Change of the RC value - AddRoundKey execution of another round (if the new $RC > 10$, AddRoundKey with unknown values) - Addition or reduction of total rounds (even repetitive execution or suppression of several rounds)
ARK	- Change of the RC value - Without any effect on the current AddRoundKey execution - Addition or reduction of total rounds (even repetitive execution or suppression of several rounds)
COMEBACK	- Change of the RC value - Addition or reduction of total rounds (even repetitive execution or suppression of several rounds)

Effects of different single-bit fault value at each of the round stages are shown in appendix A. We only report the cases which are exploitable by light differential cryptanalysis in table 4.3.

Hereafter, we use these abbreviations for the terms related to round modification attacks:

- NR : Number of total executed rounds
- R_m : A round executed identically to the middle rounds; *i.e.* including **SubBytes**, **ShiftRows**, **MixColumns** and **AddRoundKey** operations. However, the **AddRoundKey** operation uses an invalid key. The index of the searched invalid key is mentioned by “m =” expression when necessary.
- R_n : A round executed identically to the middle rounds; *i.e.* including **SubBytes**, **ShiftRows**, **MixColumns** and **AddRoundKey** operations. The **AddRoundKey** operation uses a valid key. The index of the key is always between 1 and 9. It is mentioned by “n =” expression when necessary.
- R_{n+1} or R_{n+2} : A round executed identically to the middle rounds; *i.e.* including **SubBytes**, **ShiftRows**, **MixColumns** and **AddRoundKey** operations. The **AddRoundKey** operation uses a valid key. The index of the key is always between 1 and 9. It is mentioned by “n =” expression when necessary.

- R_f : A round executed identically to the final round; i.e. including **SubBytes**, **ShiftRows** and **AddRoundKey** operations. However, the **AddRoundKey** operation uses an invalid key. The index of the searched invalid key is mentioned by “f =” expression when necessary.

In table 4.3, each filled cell refers to a possible attack. The black line above the cells shows RC value before and after the attack according to each fault model. Crossing row and column of each cell refer to the attack moment and the value of the injected fault.

Each cell contains a brief description of the corresponding attack: The first line shows the number of total executed rounds for AES encryption. This value includes all the executed rounds before and after the attack, except for the uncounted initial round.

The second line presents the list of executed rounds. For brevity, consecutive normal rounds are replaced by two points in the list, but the faulty rounds are listed separately. For instance, “ $R_0..R_8-R_f$ ” means round 0 until round 8 are executed normally, then a round similar to the final round is executed. The next one or two lines show execution condition of the faulty rounds. For the current example, the next line refers to the faulty final round; e.g. when $f=25$, the **AddRoundKey** operation of the final round searches for the subkey values corresponding to round 25.

Any round does not exist after R_{10} in AES-128. Therefore, any subkey does not exist for rounds above 10. As we described before in this chapter, our algorithm does not check if the round index is valid and will search for a corresponding bloc of 16 bytes of memory for any subkey index.

As another example, $R_0-R_1-R_m-R_f$ means that a total of 3 rounds are executed, after the initial round: Round 1 in normal execution, followed by a faulty middle round and a faulty final round. The next lines, e.g. $m=17$ and $f=18$ show that the faulty middle and final rounds search for subkey values corresponding to rounds 17 and 18.

Table 4.2 shows a brief comparison between a correct and a faulty execution.

Table 4.2: Comparison between a correct and a faulty execution.

A correct execution	Example for a round modified execution
$NR=10$	$NR \rightarrow 3$
$R_0..R_{10}$	$R_0-R_1-R_m-R_f$
	$m=17$
	$f=18$

Table 4.3: Exploitable cases of the scenario I.

Round counter	Attack moment	Model of injected fault							
		0x01	0x02	0x04	0x08	0x10	0x20	0x40	0x80
$RC = 1$		$RC \rightarrow 0$	$RC \rightarrow 3$	$RC \rightarrow 5$	$RC \rightarrow 9$	$RC \rightarrow 17$	$RC \rightarrow 33$	$RC \rightarrow 65$	$RC \rightarrow 129$
	FOR					$NR \rightarrow 1$ R_0-R_f $f=17$	$NR \rightarrow 1$ R_0-R_f $f=33$	$NR \rightarrow 1$ R_0-R_f $f=65$	$NR \rightarrow 1$ R_0-R_f $f=129$
	LOOP					$NR \rightarrow 2$ $R_0-R_m-R_f$ $m=17$ $f=18$	$NR \rightarrow 2$ $R_0-R_m-R_f$ $m=33$ $f=34$	$NR \rightarrow 2$ $R_0-R_m-R_f$ $m=65$ $f=66$	$NR \rightarrow 2$ $R_0-R_m-R_f$ $m=129$ $f=130$
	ARK COMEBACK					$NR \rightarrow 2$ $R_0-R_1-R_f$ $f=18$	$NR \rightarrow 2$ $R_0-R_1-R_f$ $f=34$	$NR \rightarrow 2$ $R_0-R_1-R_f$ $f=66$	$NR \rightarrow 2$ $R_0-R_1-R_f$ $f=130$
$RC = 2$		$RC \rightarrow 3$	$RC \rightarrow 0$	$RC \rightarrow 6$	$RC \rightarrow 10$	$RC \rightarrow 18$	$RC \rightarrow 34$	$RC \rightarrow 66$	$RC \rightarrow 130$
	FOR					$NR \rightarrow 2$ $R_0-R_1-R_f$ $f=18$	$NR \rightarrow 2$ $R_0-R_1-R_f$ $f=34$	$NR \rightarrow 2$ $R_0-R_1-R_f$ $f=66$	$NR \rightarrow 2$ $R_0-R_1-R_f$ $f=130$
	LOOP					$NR \rightarrow 3$ $R_0-R_1-R_m-R_f$ $m=18$ $f=19$	$NR \rightarrow 3$ $R_0-R_1-R_m-R_f$ $m=34$ $f=35$	$NR \rightarrow 3$ $R_0-R_1-R_m-R_f$ $m=66$ $f=67$	$NR \rightarrow 3$ $R_0-R_1-R_m-R_f$ $m=130$ $f=131$

Table 4.3 – Table continued from previous page

Round counter	Attack moment	Model of injected fault							
		0x01	0x02	0x04	0x08	0x10	0x20	0x40	0x80
$RC = 8$		$RC \rightarrow 9$	$RC \rightarrow 10$	$RC \rightarrow 12$	$RC \rightarrow 0$	$RC \rightarrow 24$	$RC \rightarrow 40$	$RC \rightarrow 72$	$RC \rightarrow 136$
	ARK COMEBACK	$NR \rightarrow 9$ $R_0 \cdot R_8 - R_f$ $f=11$	$NR \rightarrow 9$ $R_0 \cdot R_8 - R_f$ $f=13$	$NR \rightarrow 9$ $R_0 \cdot R_8 - R_f$ $f=13$		$NR \rightarrow 9$ $R_0 \cdot R_8 - R_f$ $f=25$	$NR \rightarrow 9$ $R_0 \cdot R_8 - R_f$ $f=41$	$NR \rightarrow 9$ $R_0 \cdot R_8 - R_f$ $f=73$	$NR \rightarrow 9$ $R_0 \cdot R_8 - R_f$ $f=137$
$RC = 9$		$RC \rightarrow 8$	$RC \rightarrow 11$	$RC \rightarrow 13$	$RC \rightarrow 1$	$RC \rightarrow 25$	$RC \rightarrow 41$	$RC \rightarrow 73$	$RC \rightarrow 137$
	FOR	$NR \rightarrow 9$ $R_0 \cdot R_8 - R_f$ $f=11$	$NR \rightarrow 9$ $R_0 \cdot R_8 - R_f$ $f=13$	$NR \rightarrow 9$ $R_0 \cdot R_8 - R_f$ $f=13$		$NR \rightarrow 9$ $R_0 \cdot R_8 - R_f$ $f=25$	$NR \rightarrow 9$ $R_0 \cdot R_8 - R_f$ $f=41$	$NR \rightarrow 9$ $R_0 \cdot R_8 - R_f$ $f=73$	$NR \rightarrow 9$ $R_0 \cdot R_8 - R_f$ $f=137$
	LOOP	$NR \rightarrow 10$ $R_0 \cdot R_8 - R_m - R_f$ $m=11$ $f=12$	$NR \rightarrow 10$ $R_0 \cdot R_8 - R_m - R_f$ $m=13$ $f=14$	$NR \rightarrow 10$ $R_0 \cdot R_8 - R_m - R_f$ $m=13$ $f=14$		$NR \rightarrow 10$ $R_0 \cdot R_8 - R_m - R_f$ $m=25$ $f=26$	$NR \rightarrow 10$ $R_0 \cdot R_8 - R_m - R_f$ $m=41$ $f=42$	$NR \rightarrow 10$ $R_0 \cdot R_8 - R_m - R_f$ $m=73$ $f=74$	$NR \rightarrow 10$ $R_0 \cdot R_8 - R_m - R_f$ $m=137$ $f=138$
	ARK COMEBACK	$NR \rightarrow 10$ $R_0 \cdot R_9 - R_f$ $f=12$	$NR \rightarrow 10$ $R_0 \cdot R_9 - R_f$ $f=14$	$NR \rightarrow 10$ $R_0 \cdot R_9 - R_f$ $f=14$		$NR \rightarrow 10$ $R_0 \cdot R_9 - R_f$ $f=26$	$NR \rightarrow 10$ $R_0 \cdot R_9 - R_f$ $f=42$	$NR \rightarrow 10$ $R_0 \cdot R_9 - R_f$ $f=74$	$NR \rightarrow 10$ $R_0 \cdot R_9 - R_f$ $f=138$
$RC = 10$		$RC \rightarrow 11$	$RC \rightarrow 8$	$RC \rightarrow 14$	$RC \rightarrow 2$	$RC \rightarrow 26$	$RC \rightarrow 42$	$RC \rightarrow 74$	$RC \rightarrow 138$
	FOR	$NR \rightarrow 10$ $R_0 \cdot R_9 - R_f$ $f=11$	$NR \rightarrow 10$ $R_0 \cdot R_9 - R_f$ $f=14$	$NR \rightarrow 10$ $R_0 \cdot R_9 - R_f$ $f=14$		$NR \rightarrow 10$ $R_0 \cdot R_9 - R_f$ $f=26$	$NR \rightarrow 10$ $R_0 \cdot R_9 - R_f$ $f=42$	$NR \rightarrow 10$ $R_0 \cdot R_9 - R_f$ $f=74$	$NR \rightarrow 10$ $R_0 \cdot R_9 - R_f$ $f=138$
	LOOP	$NR \rightarrow 10$ $R_0 \cdot R_9 - R_f$ $f=11$	$NR \rightarrow 10$ $R_0 \cdot R_9 - R_f$ $f=14$	$NR \rightarrow 10$ $R_0 \cdot R_9 - R_f$ $f=14$		$NR \rightarrow 10$ $R_0 \cdot R_9 - R_f$ $f=26$	$NR \rightarrow 10$ $R_0 \cdot R_9 - R_f$ $f=42$	$NR \rightarrow 10$ $R_0 \cdot R_9 - R_f$ $f=74$	$NR \rightarrow 10$ $R_0 \cdot R_9 - R_f$ $f=138$

4.2.1.2 Attacks on The Round Number Reference

The second attack scenario changes the reference number of total rounds during AES execution. This reference number is accessed only once per round and only at the beginning of any middle round and the final round. For the final round, whatever the value of the round counter is, it can never prevent the execution of the final round. However, depending on the attack, the final round might be other than R_{10} .

Table 4.4: Effects of fault injection on the round number reference during different moments of AES middle or final round execution

Instant	Effects
FOR	- Addition or reduction of total round number (even repetitive execution or suppression of several rounds) - Immediate effect of the injected fault (at the end of FOR)
LOOP ARK COMEBACK	- Addition or reduction of total round number (even suppression of several rounds but not repetitive execution) - Delayed effect of the injected fault (until beginning of the next round)

Depending on the moment of fault injection, various changes in the algorithm execution can occur. We show these effects briefly in table 4.4.

The consequences of injecting different values of faults into the round number reference are shown in Appendix A. We show in table 4.5 only the exploitable cases.

Table 4.5: The exploitable attacks of Scenario II.

Round counter	Attack moment	Model of injected fault			
		0x01	0x02	0x04	0x08
		$R_{max} \rightarrow 11$	$R_{max} \rightarrow 8$	$R_{max} \rightarrow 14$	$R_{max} \rightarrow 2$
$0 \leq RC \leq 7$					
	FOR	$NR \rightarrow 11$		$NR \rightarrow 14$	
	LOOP	$R_0 \dots R_9 - R_m - R_f$		$R_0 \dots R_9 - 4 \times R_m - R_f$	
	ARK	$m = 10$		$m = \{10, 11, 12, 13\}$	
	COMEBACK	$f = 11$		$f = 14$	
$RC = 8$					
	FOR	$NR \rightarrow 11$		$NR \rightarrow 14$	
		$R_0 \dots R_9 - R_m - R_f$		$R_0 \dots R_9 - 4 \times R_m - R_f$	
		$m = 10$		$m = \{10, 11, 12, 13\}$	
		$f = 11$		$f = 14$	

Table continued on next page

Table 4.5 – Table continued from previous page

Round counter	Attack moment	Model of injected fault			
		0x01	0x02	0x04	0x08
		$R_{max} \rightarrow 11$	$R_{max} \rightarrow 8$	$R_{max} \rightarrow 14$	$R_{max} \rightarrow 2$
	LOOP	$NR \rightarrow 11$	$NR \rightarrow 9$	$NR \rightarrow 14$	
	ARK	$R_0 \dots R_9 \cdot R_m \cdot R_f$	$R_0 \dots R_8 \cdot R_f$	$R_0 \dots R_9 \cdot 4 \times R_m \cdot R_f$	
	COMEBACK	$m = 10$ $f = 11$	$f = 9$	$m = \{10, 11, 12, 13\}$ $f = 14$	
$RC = 9$					
	FOR	$NR \rightarrow 11$	$NR \rightarrow 9$	$NR \rightarrow 14$	
		$R_0 \dots R_9 \cdot R_m \cdot R_f$	$R_0 \dots R_8 \cdot R_f$	$R_0 \dots R_9 \cdot 4 \times R_m \cdot R_f$	
		$m = 10$ $f = 11$	$f = 9$	$m = \{10, 11, 12, 13\}$ $f = 14$	
	LOOP	$NR \rightarrow 11$		$NR \rightarrow 14$	
	ARK	$R_0 \dots R_9 \cdot R_m \cdot R_f$		$R_0 \dots R_9 \cdot 4 \times R_m \cdot R_f$	
	COMEBACK	$m = 10$ $f = 11$		$m = \{10, 11, 12, 13\}$ $f = 14$	
$RC = 10$					
	FOR	$NR \rightarrow 11$		$NR \rightarrow 14$	
		$R_0 \dots R_9 \cdot R_m \cdot R_f$		$R_0 \dots R_9 \cdot 4 \times R_m \cdot R_f$	
		$m = 10$ $f = 11$		$m = \{10, 11, 12, 13\}$ $f = 14$	

4.2.2 Cryptanalysis of the Main Attacks

We saw two attack scenarios that appear feasible on our AES. According to our algorithm and our models of feasible faults, we cannot perform the Round Reduction Attack of H. Choukri and M. Tunstall. We can, however, find feasible attacks that have similarities to their cryptanalysis solution and also several new attacks with simple cryptanalysis to reveal the secret key.

Now, we describe practical solutions for exploiting two aforementioned scenarios.

4.2.2.1 Attacks on the Round Counter Value

For the first scenario, two possibilities are conceivable:

1. Attack with memory access:

If the opponent can access the memory and overwrite zero (0x00) or any known values on the addresses that are higher than the stored round keys (at least for the corresponding bytes of the attack), then we consider the attack to be *with memory access*.

Therefore, the memory content for the required non-valid round key will be equal to zero or to any desired (or known) value.

A similar possibility is when the corresponding address for a non-valid round key is out of the memory space: for instance, according to the attack, the algorithm searches for K_{129} and its corresponding address is out of the memory space. In this case, the algorithm returns a known value for such non valid and out of memory round key.

2. Attack without memory access:

If the attacker cannot overwrite memory, when the program searches for a non-valid round key (in the valid memory space), the transmitted values will be unknown.

Each of these two possibilities results in different equations and solutions. Beginning just below and for the next several pages, we describe the exploitation of these attacks of the Scenario I.

Scenario I - Attack 1

Attack moment: $RC=1$ FOR

Required fault model: $\{10, 20, 40, 80\}$

Algorithm execution: R_0-R_f

• Attack I-1 with memory access

Only one faulty ciphertext (D) is required:

$$D = \text{SR} \circ \text{SB}(M \oplus K) \quad (4.1)$$

By reversing **ShiftRow** and **SubBytes** operations in equation 4.1, we get equation 4.2:

$$\text{SB}^{-1} \circ \text{SR}^{-1}(D) = M \oplus K \quad (4.2)$$

So, we can calculate the value of K by equation 4.3:

$$K = \text{SB}^{-1} \circ \text{SR}^{-1}(D) \oplus M \quad (4.3)$$

D and M have known values.

Therefore, each byte of K is revealed by equation 4.3.

• **Attack I-1 without memory access**

For exploitation of this attack, two faulty ciphertexts (D^a and D^b) and their corresponding plaintexts (M^a and M^b) are required:

$$D^a = \text{SR} \circ \text{SB}(M^a \oplus K) \oplus K'_y \quad (4.4a)$$

$$D^b = \text{SR} \circ \text{SB}(M^b \oplus K) \oplus K'_y \quad (4.4b)$$

Each faulty ciphertext has an xor operation with K'_y which has an unknown value as an invalid key. By performing an xor operation between two faulty ciphertexts 4.4, we discard the effect of K'_y values on the resulting equation 4.5:

$$D^a \oplus D^b = \text{SR} \circ \text{SB}(M^a \oplus K) \oplus \text{SR} \circ \text{SB}(M^b \oplus K) \quad (4.5)$$

Then, we reverse **ShiftRows** operation in equation 4.5. According to **ShiftRows** distributivity property, we obtain equation 4.6:

$$\text{SR}^{-1}(D^a \oplus D^b) = \text{SB}(M^a \oplus K) \oplus \text{SB}(M^b \oplus K) \quad (4.6)$$

D^a , D^b , M^a and M^b have known values in equation 4.6.

For revealing each byte of K , we must perform a brute-force search between 2^8 possible values.

For each byte of K , the brute-force search gives two different values. Despite the fact that only one key byte value is correct, both of these two byte values satisfy equation 4.6.

When a K'_y byte is null or has a known value, the correct hypothesis for the corresponding byte of K can be examined using one of the equations 4.4. Otherwise, a new brute-force search is needed between two hypotheses for each key byte. In this case, 2^{16} entire key values must be examined in order to find the unique corresponding key.

An alternative solution is to use a third pair consisting of a plaintext and its corresponding faulty ciphertext; for instance, M_c and D_c , as shown in equation 4.7. Therefore, the brute-force search for the key byte values on both equations 4.6 and 4.8 gives the unique correct key byte value.

$$D^c = \text{SR} \circ \text{SB}(M^c \oplus K) \oplus K'_y \quad (4.7)$$

$$\text{SR}^{-1}(D^a \oplus D^c) = \text{SB}(M^a \oplus K) \oplus \text{SB}(M^c \oplus K) \quad (4.8)$$

However, some exceptions persist: When one of the byte values is repeated on the corresponding byte on all the plaintexts, the brute-force search cannot find its original value.

Scenario I - Attack 2

Attack moment: $RC=1$ LOOP

Required fault model: $\{10, 20, 40, 80\}$

Algorithm execution: $R_0-R_m-R_f$

• Attack I-2 with memory access

Only one faulty ciphertext (D) is required:

$$D = \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M \oplus K) \oplus 00] \oplus 00 \quad (4.9)$$

If we reverse **ShiftRows** and **SubBytes** operations in equation 4.9, we obtain equation 4.10:

$$\text{SB}^{-1} \circ \text{SR}^{-1}(D) = \text{MC} \circ \text{SR} \circ \text{SB}(M \oplus K) \quad (4.10)$$

By revering operations of the previous round in equation 4.10, we get the new equation 4.11:

$$\text{SB}^{-1} \circ \text{SR}^{-1} \circ \text{MC}^{-1} \circ \text{SB}^{-1} \circ \text{SR}^{-1}(D) = M \oplus K \quad (4.11)$$

Then, we can obtain equation 4.12 for finding K byte values:

$$K = \text{SB}^{-1} \circ \text{SR}^{-1} \circ \text{MC}^{-1} \circ \text{SB}^{-1} \circ \text{SR}^{-1}(D) \oplus M \quad (4.12)$$

D and M have known values.

The equation 4.12 reveals all the bytes of K .

• Attack I-2 without memory access

For exploitation of this attack, three pairs of corresponding plaintext and faulty ciphertext are required:

$$D^a = \text{SR} \circ \text{SB}\{\text{MC} \circ \text{SR} \circ \text{SB}(M^a \oplus K) \oplus K'_x\} \oplus K'_y \quad (4.13a)$$

$$D^b = \text{SR} \circ \text{SB}\{\text{MC} \circ \text{SR} \circ \text{SB}(M^b \oplus K) \oplus K'_x\} \oplus K'_y \quad (4.13b)$$

$$D^c = \text{SR} \circ \text{SB}\{\text{MC} \circ \text{SR} \circ \text{SB}(M^c \oplus K) \oplus K'_x\} \oplus K'_y \quad (4.13c)$$

Each faulty ciphertext, shown as an equation 4.13, has two invalid key values. This attack results in the execution of a middle round with an invalid key value, namely K'_x , followed by the final round with another invalid key value, namely K'_y .

We can omit the effect of K'_y by performing an xor operation between the first faulty ciphertext and each of other ones:

$$\begin{aligned} D^a \oplus D^b &= \text{SR} \circ \text{SB}\{\text{MC} \circ \text{SR} \circ \text{SB}(M^a \oplus K) \oplus K'_x\} \oplus \\ &\quad \text{SR} \circ \text{SB}\{\text{MC} \circ \text{SR} \circ \text{SB}(M^b \oplus K) \oplus K'_x\} \end{aligned} \quad (4.14a)$$

$$\begin{aligned} D^a \oplus D^c &= \text{SR} \circ \text{SB}\{\text{MC} \circ \text{SR} \circ \text{SB}(M^a \oplus K) \oplus K'_x\} \oplus \\ &\quad \text{SR} \circ \text{SB}\{\text{MC} \circ \text{SR} \circ \text{SB}(M^c \oplus K) \oplus K'_x\} \end{aligned} \quad (4.14b)$$

Nevertheless, K'_x remains in the equations 4.14. The solution for resolving the equations 4.14 consists in creating hypotheses on K'_x value, in addition to K value.

M^a , M^b , M^c , D^a , D^b and D^c have known values.

So, a brute-force search of $(2^8)^4 \times 2^8 \times 4 = 2^{42}$ values is necessary for each column of K . It results in only one hypothesis for the corresponding column of K and also K'_x . The entire key is discovered after a brute-force search of $2^{42} \times 4 = 2^{44}$ values.

Scenario I - Attack 3

Attack moment: $RC=1$ ARK, COMEBACK or $RC=2$ FOR

Required fault model: $\{10, 20, 40, 80\}$

Algorithm execution: R_0 - R_1 - R_f

• Attack I-3 with memory access

Two faulty ciphertexts (D^a and D^b) are required:

$$D^a = \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M^a \oplus K) \oplus K_1] \oplus 00 \quad (4.15a)$$

$$D^b = \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M^b \oplus K) \oplus K_1] \oplus 00 \quad (4.15b)$$

By reversing **ShiftRows** and **SubBytes** operations in equation 4.15, we obtain equation 4.16:

$$\text{SB}^{-1} \circ \text{SR}^{-1}(D^a) = \text{MC} \circ \text{SR} \circ \text{SB}(M^a \oplus K) \oplus K_1 \quad (4.16a)$$

$$\text{SB}^{-1} \circ \text{SR}^{-1}(D^b) = \text{MC} \circ \text{SR} \circ \text{SB}(M^b \oplus K) \oplus K_1 \quad (4.16b)$$

Then, an xor operation between two equations 4.16 removes K_1 from the equations and gives equation 4.17:

$$\begin{aligned} &\text{SB}^{-1} \circ \text{SR}^{-1}(D^a) \oplus \text{SB}^{-1} \circ \text{SR}^{-1}(D^b) = \\ &\text{MC} \circ \text{SR} \circ \text{SB}(M^a \oplus K) \oplus \text{MC} \circ \text{SR} \circ \text{SB}(M^b \oplus K) \end{aligned} \quad (4.17)$$

By reversing `MixColumns` and `ShiftRows` operations and according to their distributivity property, we obtain:

$$\begin{aligned} \text{SR}^{-1} \circ \text{MC}^{-1} [\text{SB}^{-1} \circ \text{SR}^{-1}(D^a) \oplus \text{SB}^{-1} \circ \text{SR}^{-1}(D^b)] = \\ \text{SB}(M^a \oplus K) \oplus \text{SB}(M^b \oplus K) \end{aligned} \quad (4.18)$$

D^a , D^b , M^a and M^b have known values.

So, we can perform a brute force search for each byte of K separately. For each byte of K , we have 2^8 possible values to check. This brute force search results in two hypotheses for each byte of K .

Although only one key byte value is correct, both of these two byte values satisfy equation 4.18. So, a new brute-force search is needed between two hypotheses for each key byte. In this case, for each key hypothesis, K_1 also must be calculated by `KeyExpansion` operations and then examined using one of the equations 4.15. Therefore, K_1 expansion is needed for 2^{16} entire key hypotheses in order to find the unique corresponding key.

An alternative solution is to use a third pair of plaintext and its corresponding faulty ciphertext; for instance: M_c and D_c , shown in equation 4.19. So, the brute-force search for the key byte values on both equations 4.24 and 4.20 gives only the correct key byte value.

$$D^c = \text{SR} \circ \text{SB} [\text{MC} \circ \text{SR} \circ \text{SB}(M^c \oplus K) \oplus K_1] \oplus 00 \quad (4.19)$$

$$\begin{aligned} \text{SR}^{-1} \circ \text{MC}^{-1} [\text{SB}^{-1} \circ \text{SR}^{-1}(D^a) \oplus \text{SB}^{-1} \circ \text{SR}^{-1}(D^b)] = \\ \text{SB}(M^a \oplus K) \oplus \text{SB}(M^b \oplus K) \end{aligned} \quad (4.20)$$

• Attack I-3 without memory access

For finding the key using this attack, 3 pairs of corresponding faulty ciphertext and original plaintext are necessary.

Equations 4.15 show two first faulty ciphertexts. Each faulty ciphertext has K'_y , as an invalid key value:

$$D^a = \text{SR} \circ \text{SB} [\text{MC} \circ \text{SR} \circ \text{SB}(M^a \oplus K) \oplus K_1] \oplus K'_y \quad (4.21a)$$

$$D^b = \text{SR} \circ \text{SB} [\text{MC} \circ \text{SR} \circ \text{SB}(M^b \oplus K) \oplus K_1] \oplus K'_y \quad (4.21b)$$

$$D^c = \text{SR} \circ \text{SB} [\text{MC} \circ \text{SR} \circ \text{SB}(M^c \oplus K) \oplus K_1] \oplus K'_y \quad (4.21c)$$

According to xor substitution property, we can move K'_y in equations 4.21 and obtain equations 4.22:

$$D^a \oplus K'_y = \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M^a \oplus K) \oplus K_1] \quad (4.22a)$$

$$D^b \oplus K'_y = \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M^b \oplus K) \oplus K_1] \quad (4.22b)$$

$$D^c \oplus K'_y = \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M^c \oplus K) \oplus K_1] \quad (4.22c)$$

By reversing **ShiftRows** and **SubBytes** in equations 4.22, we obtain equations 4.23:

$$\text{SB}^{-1} \circ \text{SR}^{-1}(D^a \oplus K'_y) = \text{MC} \circ \text{SR} \circ \text{SB}(M^a \oplus K) \oplus K_1 \quad (4.23a)$$

$$\text{SB}^{-1} \circ \text{SR}^{-1}(D^b \oplus K'_y) = \text{MC} \circ \text{SR} \circ \text{SB}(M^b \oplus K) \oplus K_1 \quad (4.23b)$$

$$\text{SB}^{-1} \circ \text{SR}^{-1}(D^c \oplus K'_y) = \text{MC} \circ \text{SR} \circ \text{SB}(M^c \oplus K) \oplus K_1 \quad (4.23c)$$

We can discard K_1 value from equations 4.23 by an xor operation between them and we arrive at equations 4.24:

$$\begin{aligned} \text{SB}^{-1} \circ \text{SR}^{-1}(D^a \oplus K'_y) \oplus \text{SB}^{-1} \circ \text{SR}^{-1}(D^b \oplus K'_y) = \\ \text{MC} \circ \text{SR} \circ \text{SB}(M^a \oplus K) \oplus \text{MC} \circ \text{SR} \circ \text{SB}(M^b \oplus K) \end{aligned} \quad (4.24a)$$

$$\begin{aligned} \text{SB}^{-1} \circ \text{SR}^{-1}(D^a \oplus K'_y) \oplus \text{SB}^{-1} \circ \text{SR}^{-1}(D^c \oplus K'_y) = \\ \text{MC} \circ \text{SR} \circ \text{SB}(M^a \oplus K) \oplus \text{MC} \circ \text{SR} \circ \text{SB}(M^c \oplus K) \end{aligned} \quad (4.24b)$$

D^a, D^b, D^c, M^a, M^b and M^c have known values.

We create hypotheses by a brute-force search on each quadruple of K'_y bytes that enters together in a **MixColumns** operation. Then, for each hypothesis on a K'_y quadruple, we search values for the 4 corresponding bytes of K . As there are no **MixColumns** steps for K values in equation 4.24, we can perform separately our brute-force search for each of the bytes on the quadruple.

In order to find the unique values for a K'_y quadruple, we must search and verify all the $(2^8)^4$ values. In addition, finding each byte of a K quadruple requires a brute-force search among all the 2^8 values. So, an exhaustive search of 2^{42} values is necessary in order to reveal a K quadruple or 2^{44} values for the full key.

Scenario I - Attack 4

Attack moment: $RC=2$ LOOP

Required fault model: $\{10, 20, 40, 80\}$

Algorithm execution: $R_0-R_1-R_m-R_f$

• **Attack I-4 with memory access**

Two faulty ciphertexts (D^a and D^b) and their corresponding plaintexts are required:

$$D^a = \text{SR} \circ \text{SB}\{\text{MC} \circ \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M^a \oplus K) \oplus K_1] \oplus 00\} \oplus 00 \quad (4.25a)$$

$$D^b = \text{SR} \circ \text{SB}\{\text{MC} \circ \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M^b \oplus K) \oplus K_1] \oplus 00\} \oplus 00 \quad (4.25b)$$

By reversing **ShiftRows** and **SubBytes** operations in equations 4.25, we obtain equations 4.26:

$$\text{SB}^{-1} \circ \text{SR}^{-1}(D^a) = \text{MC} \circ \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M^a \oplus K) \oplus K_1] \quad (4.26a)$$

$$\text{SB}^{-1} \circ \text{SR}^{-1}(D^b) = \text{MC} \circ \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M^b \oplus K) \oplus K_1] \quad (4.26b)$$

Then, by reversing the operations of the previous round in equations 4.26, we get equations 4.27:

$$\text{SB}^{-1} \circ \text{SR}^{-1} \circ \text{MC}^{-1} \circ \text{SB}^{-1} \circ \text{SR}^{-1}(D^a) = \text{MC} \circ \text{SR} \circ \text{SB}(M^a \oplus K) \oplus K_1 \quad (4.27a)$$

$$\text{SB}^{-1} \circ \text{SR}^{-1} \circ \text{MC}^{-1} \circ \text{SB}^{-1} \circ \text{SR}^{-1}(D^b) = \text{MC} \circ \text{SR} \circ \text{SB}(M^b \oplus K) \oplus K_1 \quad (4.27b)$$

Afterwards, we can discard the effect of K_1 in equations 4.27 by an xor operation on the two equations. Then, we obtain equation 4.28:

$$\begin{aligned} & \text{SB}^{-1} \circ \text{SR}^{-1} \circ \text{MC}^{-1} \circ \text{SB}^{-1} \circ \text{SR}^{-1}(D^a) \oplus \\ & \text{SB}^{-1} \circ \text{SR}^{-1} \circ \text{MC}^{-1} \circ \text{SB}^{-1} \circ \text{SR}^{-1}(D^b) = \\ & \text{MC} \circ \text{SR} \circ \text{SB}(M^a \oplus K) \oplus \text{MC} \circ \text{SR} \circ \text{SB}(M^b \oplus K) \end{aligned} \quad (4.28)$$

By reversing **MixColumns** and **ShiftRows** operations in the equation 4.28 and according to their distributivity property, we obtain the new equation 4.29:

$$\begin{aligned} & \text{SR}^{-1} \circ \text{MC}^{-1}[\text{SB}^{-1} \circ \text{SR}^{-1} \circ \text{MC}^{-1} \circ \text{SB}^{-1} \circ \text{SR}^{-1}(D^a) \oplus \\ & \text{SB}^{-1} \circ \text{SR}^{-1} \circ \text{MC}^{-1} \circ \text{SB}^{-1} \circ \text{SR}^{-1}(D^b)] = \\ & \text{SB}(M^a \oplus K) \oplus \text{SB}(M^b \oplus K) \end{aligned} \quad (4.29)$$

D^a , D^b , M^a and M^b have known values.

So, for each column of 4 bytes on K in equation 4.29, a brute-force search among $(2^8)^4$ values reveals the corresponding bytes of K . Therefore, a brute-force search between 2^{34} values reveals the entire key.

- **Attack I-4 without memory access**

Each faulty ciphertext, such as D^a and D^b in equations 4.30, has K'_x and K'_y which are two invalid key values.

$$D^a = \text{SR} \circ \text{SB}\{\text{MC} \circ \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M^a \oplus K) \oplus K_1] \oplus K'_x\} \oplus K'_y \quad (4.30a)$$

$$D^b = \text{SR} \circ \text{SB}\{\text{MC} \circ \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M^b \oplus K) \oplus K_1] \oplus K'_x\} \oplus K'_y \quad (4.30b)$$

By performing an xor operation on the two faulty ciphertexts of equations 4.30, we can eliminate K'_y and obtain equation 4.31:

$$\begin{aligned} D^a \oplus D^b = \\ \text{SR} \circ \text{SB}\{\text{MC} \circ \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M^a \oplus K) \oplus K_1] \oplus K'_x\} \oplus \\ \text{SR} \circ \text{SB}\{\text{MC} \circ \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M^b \oplus K) \oplus K_1] \oplus K'_x\} \end{aligned} \quad (4.31)$$

K'_x cannot be excluded from equation 4.31. Searching for both K and K_1 , in the presence of the invalid key values of K'_x and in the presence of the two **MixColumns** steps cannot be reduced to a smaller brute-force search. Therefore, in this attack, without access to the memory, K cannot be discovered in a reasonable time.

Scenario I - Attack 5

Attack moment: $RC=8$ ARK, COMEBACK

Required fault model: $\{01\}$

Algorithm execution: $R_0-R_1..R_8-R_{10}$

- **Attack I-5 with or without memory access**

For exploiting this attack, at least two pairs of corresponding faulty and correct ciphertexts are required:

$$C^a = \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M_8^a) \oplus K_9] \oplus K_{10} \quad (4.32a)$$

$$C^b = \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M_8^b) \oplus K_9] \oplus K_{10} \quad (4.32b)$$

$$D^a = \text{SR} \circ \text{SB}(M_8^a) \oplus K_{10} \quad (4.33a)$$

$$D^b = \text{SR} \circ \text{SB}(M_8^b) \oplus K_{10} \quad (4.33b)$$

According to the substitution property of xor, we can move K_{10} in equations 4.32 and obtain equations 4.34:

$$C^a \oplus K_{10} = \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M_8^a) \oplus K_9] \quad (4.34a)$$

$$C^b \oplus K_{10} = \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M_8^b) \oplus K_9] \quad (4.34b)$$

Then, by reversing **ShiftRows** and **SubBytes** operations in equations 4.34, we obtain equation 4.35:

$$\text{SB}^{-1} \circ \text{SR}^{-1}(C^a \oplus K_{10}) = \text{MC} \circ \text{SR} \circ \text{SB}(M_8^a) \oplus K_9 \quad (4.35a)$$

$$\text{SB}^{-1} \circ \text{SR}^{-1}(C^b \oplus K_{10}) = \text{MC} \circ \text{SR} \circ \text{SB}(M_8^b) \oplus K_9 \quad (4.35b)$$

We can exclude the K_9 value from the equations 4.35 by performing an xor operation on them:

$$\begin{aligned} \text{SB}^{-1} \circ \text{SR}^{-1}(C^a \oplus K_{10}) \oplus \text{SB}^{-1} \circ \text{SR}^{-1}(C^b \oplus K_{10}) = \\ \text{MC} \circ \text{SR} \circ \text{SB}(M_8^a) \oplus \text{MC} \circ \text{SR} \circ \text{SB}(M_8^b) \end{aligned} \quad (4.36)$$

Afterwards, we factorize the **MixColumns** operation according to its distributivity property in equation 4.36. In addition, we replace D^a and D^b by their corresponding values in equations 4.33 and write the new equation 4.37:

$$\text{SB}^{-1} \circ \text{SR}^{-1}(C^a \oplus K_{10}) \oplus \text{SB}^{-1} \circ \text{SR}^{-1}(C^b \oplus K_{10}) = \text{MC}(D^a \oplus D^b) \quad (4.37)$$

C^a , C^b , D^a and D^b have known values.

So, we can calculate K_{10} by the equation 4.37. By using two pairs of corresponding faulty and correct ciphertexts, this equation leads to two different values for each byte of K by an exhaustive search. Therefore, we must perform a second brute-force search between all the K_{10} byte hypotheses in order to find the correct one. In this case, for each of 2^{16} key hypotheses, we must calculate all the round keys by the reverse of **KeyExpansion** operations.

Besides, an alternative solution exists by using an additional pair of corresponding faulty and correct ciphertexts, such as C^c and D^c in equations 4.38 and 4.39.

$$C^c = \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M_8^c) \oplus K_9] \oplus K_{10} \quad (4.38)$$

$$D^c = \text{SR} \circ \text{SB}(M_8^c) \oplus K_{10} \quad (4.39)$$

Therefore, equations 4.37 and 4.40 lead to a unique value for each K_{10} byte (with some exceptions).

$$\text{SB}^{-1} \circ \text{SR}^{-1}(C^a \oplus K_{10}) \oplus \text{SB}^{-1} \circ \text{SR}^{-1}(C^c \oplus K_{10}) = \text{MC}(D^a \oplus D^c) \quad (4.40)$$

These operations are independent from any invalid key value. So, they don't require any memory access.

Scenario I - Attack 6

Attack moment: $RC=8$ ARK, COMEBACK or $RC=9$ FOR

Required fault model: $\{02, 04, 10, 20, 40, 80\}$

Algorithm execution: $R_0-R_1..R_8-R_f$

- **Attack I-6 with memory access**

For exploiting this attack, at least two pairs of corresponding faulty and correct ciphertexts are required:

$$C^a = \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M_8^a) \oplus K_9] \oplus K_{10} \quad (4.41a)$$

$$C^b = \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M_8^b) \oplus K_9] \oplus K_{10} \quad (4.41b)$$

$$D^a = \text{SR} \circ \text{SB}(M_8^a) \oplus 00 \quad (4.42a)$$

$$D^b = \text{SR} \circ \text{SB}(M_8^b) \oplus 00 \quad (4.42b)$$

By replacing D_a and D_b values from equations 4.42 in equations 4.41, we obtain the new equations 4.43:

$$C^a = \text{SR} \circ \text{SB}[\text{MC}(D_8^a) \oplus K_9] \oplus K_{10} \quad (4.43a)$$

$$C^b = \text{SR} \circ \text{SB}[\text{MC}(D_8^b) \oplus K_9] \oplus K_{10} \quad (4.43b)$$

Then, by an xor operation on the two equations 4.43, we get equation 4.44:

$$C^a \oplus C^b = \text{SR} \circ \text{SB}[\text{MC}(D_8^a) \oplus K_9] \oplus \text{SR} \circ \text{SB}[\text{MC}(D_8^b) \oplus K_9] \quad (4.44)$$

C^a , C^b , D^a and D^b have known values.

So, by replacing the values of C^a , C^b , D^a and D^b in equation 4.44, we can find each byte of K_9 separately.

By using two pairs of corresponding faulty and correct ciphertexts, solving equation 4.44 presents us with two hypotheses on each byte of K_9 . Then, a brute-force search between 2^{16} hypotheses of possible key values is necessary in order to find the corresponding K_9 .

Besides, by using a third pair of corresponding faulty and correct ciphertexts, such as C^c and D^c (shown as equations 4.45 and 4.46), we can solve both equations 4.44 and 4.47. This leads to a single value for each byte of K_9 .

$$C^c = \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M_8^c) \oplus K_9] \oplus K_{10} \quad (4.45)$$

$$D^c = \text{SR} \circ \text{SB}(M_8^c) \oplus 00 \quad (4.46)$$

$$C^a \oplus C^c = \text{SR} \circ \text{SB}[\text{MC}(D_8^a) \oplus K_9] \oplus \text{SR} \circ \text{SB}[\text{MC}(D_8^c) \oplus K_9] \quad (4.47)$$

However, in some exceptional cases, even by using 3 pairs of corresponding faulty and correct ciphertexts, more than one byte could be found for several bytes of K_9 .

• **Attack I-6 without memory access**

For exploiting this attack without access to the memory, three pairs of corresponding faulty and correct ciphertexts are required:

$$C^a = \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M_8^a) \oplus K_9] \oplus K_{10} \quad (4.48a)$$

$$C^b = \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M_8^b) \oplus K_9] \oplus K_{10} \quad (4.48b)$$

$$C^c = \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M_8^c) \oplus K_9] \oplus K_{10} \quad (4.48c)$$

$$D^a = \text{SR} \circ \text{SB}(M_8^a) \oplus K'_y \quad (4.49a)$$

$$D^b = \text{SR} \circ \text{SB}(M_8^b) \oplus K'_y \quad (4.49b)$$

$$D^c = \text{SR} \circ \text{SB}(M_8^c) \oplus K'_y \quad (4.49c)$$

Equations 4.49 have the value of the invalid key K'_y . So, by performing xor operations on the first faulty ciphertext (D^a) and the following ones, we discard K'_y and we obtain equations 4.50:

$$D^a \oplus D^b = \text{SR} \circ \text{SB}(M_8^a) \oplus \text{SR} \circ \text{SB}(M_8^b) \quad (4.50a)$$

$$D^a \oplus D^c = \text{SR} \circ \text{SB}(M_8^a) \oplus \text{SR} \circ \text{SB}(M_8^c) \quad (4.50b)$$

Besides, xor operations between the first correct ciphertext (C^a) and the following ones lead to equations 4.51:

$$\begin{aligned} C^a \oplus C^b &= \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M_8^a) \oplus K_9] \oplus \\ &\quad \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M_8^b) \oplus K_9] \end{aligned} \quad (4.51a)$$

$$\begin{aligned} C^a \oplus C^c &= \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M_8^a) \oplus K_9] \oplus \\ &\quad \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M_8^c) \oplus K_9] \end{aligned} \quad (4.51b)$$

For brevity, in writing the equations, we define X^a , X^b and X^c with given definitions in equations 4.52:

$$X^a = \text{SR} \circ \text{SB}(M_8^a) \quad (4.52a)$$

$$X^b = \text{SR} \circ \text{SB}(M_8^b) \quad (4.52b)$$

$$X^c = \text{SR} \circ \text{SB}(M_8^c) \quad (4.52c)$$

Therefore, we replace X^a , X^b and X^c values in equations 4.50 and 4.51 and we obtain equations 4.53 and 4.54:

$$D^a \oplus D^b = X^a \oplus X^b \quad (4.53a)$$

$$D^a \oplus D^c = X^a \oplus X^c \quad (4.53b)$$

$$C^a \oplus C^b = \text{SR} \circ \text{SB}[\text{MC}(X^a) \oplus K_9] \oplus \text{SR} \circ \text{SB}[\text{MC}(X^b) \oplus K_9] \quad (4.54a)$$

$$C^a \oplus C^c = \text{SR} \circ \text{SB}[\text{MC}(X^a) \oplus K_9] \oplus \text{SR} \circ \text{SB}[\text{MC}(X^c) \oplus K_9] \quad (4.54b)$$

Furthermore, we rewrite X^a , X^b and X^c according to their corresponding values in equations 4.48:

$$X^a = \text{MC}^{-1}[\text{SB}^{-1} \circ \text{SR}^{-1}(C^a \oplus K_{10}) \oplus K_9] \quad (4.55a)$$

$$X^b = \text{MC}^{-1}[\text{SB}^{-1} \circ \text{SR}^{-1}(C^b \oplus K_{10}) \oplus K_9] \quad (4.55b)$$

$$X^c = \text{MC}^{-1}[\text{SB}^{-1} \circ \text{SR}^{-1}(C^c \oplus K_{10}) \oplus K_9] \quad (4.55c)$$

So, if we perform xor operations between X^a and X^b and also between X^a and X^c in equations 4.55, we obtain the new equations 4.56:

$$X^a \oplus X^b = \text{MC}^{-1}[\text{SB}^{-1} \circ \text{SR}^{-1}(C^a \oplus K_{10}) \oplus \text{SB}^{-1} \circ \text{SR}^{-1}(C^b \oplus K_{10})] \quad (4.56a)$$

$$X^a \oplus X^c = \text{MC}^{-1}[\text{SB}^{-1} \circ \text{SR}^{-1}(C^b \oplus K_{10}) \oplus \text{SB}^{-1} \circ \text{SR}^{-1}(C^c \oplus K_{10})] \quad (4.56b)$$

If we factorize the reverse of **ShiftRows** operation in equations 4.56, we get equations 4.57:

$$X^a \oplus X^b = \text{MC}^{-1} \circ \text{SR}^{-1}[\text{SB}^{-1}(C^a \oplus K_{10}) \oplus \text{SB}^{-1}(C^b \oplus K_{10})] \quad (4.57a)$$

$$X^a \oplus X^c = \text{MC}^{-1} \circ \text{SR}^{-1}[\text{SB}^{-1}(C^b \oplus K_{10}) \oplus \text{SB}^{-1}(C^c \oplus K_{10})] \quad (4.57b)$$

Then, we perform again the reverse of **ShiftRows** and **MixColumns** operations in equations 4.57. We also replace X^a , X^b and X^c according to their corresponding values in equations 4.53. So, we get new equations 4.58:

$$\text{MC} \circ \text{SR}(D^a \oplus D^b) = \text{SB}^{-1}(C^a \oplus K_{10}) \oplus \text{SB}^{-1}(C^b \oplus K_{10}) \quad (4.58a)$$

$$\text{MC} \circ \text{SR}(D^a \oplus D^c) = \text{SB}^{-1}(C^b \oplus K_{10}) \oplus \text{SB}^{-1}(C^c \oplus K_{10}) \quad (4.58b)$$

C^a, C^b, C^c, D^a, D^b and D^c have known values.

Therefore, we replace C^a, C^b, C^c, D^a, D^b and D^c values in equations 4.58. Then, by an exhaustive search, we find hypotheses on K_{10} byte values. This solution often gives one value for each byte of K_{10} . Nevertheless, in exceptional cases, it could give two hypotheses for several bytes of K_{10} . The non-corresponding hypotheses can be identified and eliminated by a new exhaustive search among the hypotheses. In this case, all the round keys must be calculated for each entire key hypothesis.

Scenario I - Attack 7

Attack moment: $RC=9$ LOOP

Required fault model: $\{02, 04, 10, 20, 40, 80\}$

Algorithm execution: $R_0-R_1..R_8-R_m-R_f$

• Attack I-7 with memory access

For this attack, two pairs of corresponding faulty and correct ciphertexts are required:

$$C^a = \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M_8^a) \oplus K_9] \oplus K_{10} \quad (4.59a)$$

$$C^b = \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M_8^b) \oplus K_9] \oplus K_{10} \quad (4.59b)$$

$$D^a = \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M_8^a) \oplus 00] \oplus 00 \quad (4.60a)$$

$$D^b = \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M_8^b) \oplus 00] \oplus 00 \quad (4.60b)$$

An xor operation between the two equations of the correct ciphertexts in 4.59, eliminates K_{10} and leads to equation 4.61:

$$\begin{aligned} C^a \oplus C^b &= \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M_8^a) \oplus K_9] \oplus \\ &\quad \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M_8^b) \oplus K_9] \end{aligned} \quad (4.61)$$

Furthermore, by reversing **ShiftRows** and **SubBytes** in equations 4.60, we get new equations 4.62:

$$\text{SB}^{-1} \circ \text{SR}^{-1}(D^a) = \text{MC} \circ \text{SR} \circ \text{SB}(M_8^a) \quad (4.62a)$$

$$\text{SB}^{-1} \circ \text{SR}^{-1}(D^b) = \text{MC} \circ \text{SR} \circ \text{SB}(M_8^b) \quad (4.62b)$$

We replace $\text{MC} \circ \text{SR} \circ \text{SB}(M_8^a)$ and $\text{MC} \circ \text{SR} \circ \text{SB}(M_8^b)$ in equation 4.61 with the corresponding values coming from equations 4.62. Therefore, we obtain new equation 4.63:

$$C^a \oplus C^b = \text{SR}\{\text{SB}[\text{SB}^{-1} \circ \text{SR}^{-1}(D^a) \oplus K_9] \oplus \text{SB}[\text{SB}^{-1} \circ \text{SR}^{-1}(D^b) \oplus K_9]\} \quad (4.63)$$

C^a , C^b , D^a and D^b have known values.

We replace C^a , C^b , D^a and D^b in equation 4.63. Then, by using an exhaustive search on K_9 , we obtain the corresponding value of each byte.

Solving the equation 4.63 often gives only one value for each byte of K_9 . However, in exceptional cases, it might result in two hypotheses for some K_9 bytes. The non corresponding values can be identified and discarded by a second exhaustive search limited to the new hypotheses.

• Attack I-7 without memory access

For successful exploitation of this attack without memory access, three pairs of corresponding faulty and correct ciphertexts are required:

$$C^a = \text{SR} \circ \text{SB}(M_9^a) \oplus K_{10} = \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M_8^a) \oplus K_9] \oplus K_{10} \quad (4.64a)$$

$$C^b = \text{SR} \circ \text{SB}(M_9^b) \oplus K_{10} = \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M_8^b) \oplus K_9] \oplus K_{10} \quad (4.64b)$$

$$C^c = \text{SR} \circ \text{SB}(M_9^c) \oplus K_{10} = \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M_8^c) \oplus K_9] \oplus K_{10} \quad (4.64c)$$

$$D^a = \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M_8^a) \oplus K'_x] \oplus K'_y \quad (4.65a)$$

$$D^b = \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M_8^b) \oplus K'_x] \oplus K'_y \quad (4.65b)$$

$$D^c = \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M_8^c) \oplus K'_x] \oplus K'_y \quad (4.65c)$$

In this attack, K'_x and K'_y are unknown values corresponding to invalid round keys. For greater simplicity, we assume K'_x as a faulty K_9 . We also assume E_9 as the value of the fault on K_9 that has changed it to K'_x , as shown in equation 4.66:

$$K'_x = K_9 \oplus E_9 \quad (4.66)$$

Therefore, we replace K'_x values in equations 4.65 according to the equation 4.66 and we get new equations 4.67:

$$D^a = \text{SR} \circ \text{SB}[(M_9^a) \oplus E_9] \oplus K'_y \quad (4.67a)$$

$$D^b = \text{SR} \circ \text{SB}[(M_9^b) \oplus E_9] \oplus K'_y \quad (4.67b)$$

$$D^c = \text{SR} \circ \text{SB}[(M_9^c) \oplus E_9] \oplus K'_y \quad (4.67c)$$

We perform xor operations between the first faulty ciphertext and each of the following ones in equations 4.67. Then, we factorize the reverse of **ShiftRows** operations. Therefore, we obtain equations 4.68:

$$\text{SR}^{-1}(D^a \oplus D^b) = \text{SB}[(M_9^a) \oplus E_9] \oplus \text{SB}[(M_9^b) \oplus E_9] \quad (4.68a)$$

$$\text{SR}^{-1}(D^a \oplus D^c) = \text{SB}[(M_9^a) \oplus E_9] \oplus \text{SB}[(M_9^c) \oplus E_9] \quad (4.68b)$$

Furthermore, we perform xor operations between the first correct ciphertext and each of the other ones in equations 4.64. We also factorize the reverse of **ShiftRows** operations and we get equations 4.69:

$$C^a \oplus C^b = \text{SR} \circ \text{SB}(M_9^a) \oplus \text{SR} \circ \text{SB}(M_9^b) \quad (4.69a)$$

$$C^a \oplus C^c = \text{SR} \circ \text{SB}(M_9^a) \oplus \text{SR} \circ \text{SB}(M_9^c) \quad (4.69b)$$

Besides, by using equations 4.69, we write M_9^b and M_9^c values as functions of M_9^a , K_9 , K_{10} and correct ciphertexts. Thus we obtain equations 4.70:

$$M_9^b = \text{SB}^{-1}[\text{SR}^{-1}(C^a \oplus C^b) \oplus \text{SB}(M_9^a)] \quad (4.70a)$$

$$M_9^c = \text{SB}^{-1}[\text{SR}^{-1}(C^a \oplus C^c) \oplus \text{SB}(M_9^a)] \quad (4.70b)$$

We replace the values for M_9^b and M_9^c in equations 4.68 by the obtained values coming from equations by equations 4.70. Thus, we get equations 4.71:

$$\begin{aligned} \text{SR}^{-1}(D^a \oplus D^b) &= \text{SB}[(M_9^a) \oplus E_9] \oplus \\ &\text{SB}[\text{SB}^{-1}(\text{SR}^{-1}(C^a \oplus C^b) \oplus \text{SB}(M_9^a)) \oplus E_9] \end{aligned} \quad (4.71a)$$

$$\begin{aligned} \text{SR}^{-1}(D^a \oplus D^c) &= \text{SB}[(M_9^a) \oplus E_9] \oplus \\ &\text{SB}[\text{SB}^{-1}(\text{SR}^{-1}(C^a \oplus C^c) \oplus \text{SB}(M_9^a)) \oplus E_9] \end{aligned} \quad (4.71b)$$

C^a , C^b , C^c , D^a , D^b and D^c have known values.

Finally, we perform an exhaustive search among 2^8 possible values for each M_9^a byte and among 2^8 possible values for each corresponding E_9 byte. This exhaustive search often leads to a unique value for each M_9^a byte and another unique value for the corresponding E_9 byte. Then, by using these M_9^a byte values and using the equation 4.72, we find K_{10} byte values. The equation 4.72 is calculated from the first correct ciphertext equation 4.64.

$$K_{10} = \text{SR} \circ \text{SB}(M_9^a) \oplus C^a \quad (4.72)$$

Scenario I - Attack 8

Attack moment: $RC=9$ ARK, COMEBACK

Required fault model: $\{02, 04, 10, 20, 40, 80\}$

Algorithm execution: $R_0-R_1..R_9-R_f$

• Attack I-8 with memory access

For exploiting this attack, only one pair of corresponding faulty and correct ciphertexts is required:

$$C = \text{SR} \circ \text{SB}(M^9) \oplus K_{10} \quad (4.73)$$

$$D = \text{SR} \circ \text{SB}(M^9) \oplus 00 \quad (4.74)$$

We can obtain K_{10} value by performing only an xor operation between equation 4.73 and equation 4.74:

$$K_{10} = C \oplus D \quad (4.75)$$

C and D have known values.

Each byte of K_{10} is revealed by the equation 4.75.

• Attack I-8 without memory access

We show a correct ciphertext and a corresponding faulty ciphertext of this attack in equations 4.76 and 4.77 respectively. K_y' value is unknown and cannot be revealed by any operation between correct and faulty ciphertexts. Hence, there is no possibility of finding K_{10} or any other key values. This attack is not exploitable without memory access.

$$C = \text{SR} \circ \text{SB}(M^9) \oplus K_{10} \quad (4.76)$$

$$D = \text{SR} \circ \text{SB}(M^9) \oplus K_y' \quad (4.77)$$

4.2.2.2 Attacks on the Round Number Reference

For the second scenario, again, both attack possibilities, with or without memory access, can be considered. However, similar to the first scenario, some independent attacks also exist.

Each of these possibilities has different consequences for the attacks and requires particular solutions. Here, we describe a cryptanalytic solution for each attack.

In the upcoming attack exploitations, sometimes two compared ciphertexts have a difference of several rounds. For greater ease, we assume that `MiddleRound` and `FinalRound` are two functions that combine different transformations of a middle and of a final round, respectively. We present them as equations 4.78 and 4.79:

$$\text{MiddleRound}(M) \text{ or } \text{MR}(M) = \text{MC} \circ \text{SR} \circ \text{SB}(M) \quad (4.78)$$

$$\text{FinalRound}(M) \text{ or } \text{FR}(M) = \text{SR} \circ \text{SB}(M) \quad (4.79)$$

The reversed functions of `MiddleRound` and `FinalRound` also represent the reverse of round operations, as shown in equations 4.80 and 4.81:

$$\text{MiddleRound}^{-1}(D) \text{ or } \text{MR}^{-1}(D) = \text{SB}^{-1} \circ \text{SR}^{-1} \circ \text{MC}^{-1}(D) \quad (4.80)$$

$$\text{FinalRound}^{-1}(C) \text{ or } \text{FR}^{-1}(C) = \text{SB}^{-1} \circ \text{SR}^{-1}(C) \quad (4.81)$$

Several times, we will use `MiddleRound` and `FinalRound` operations in upcoming attack exploitations.

Scenario II - Attack 1

Attack moment: Any time between $RC=0$ and $RC=10$ FOR

Required fault model: $\{01\}$

Algorithm execution: $R_0..R_9-R_{m=10}-R_{f=11}$

• Attack II-1 with memory access

For exploiting this attack, only one pair of corresponding faulty and correct ciphertexts is required:

$$C = \text{SR} \circ \text{SB}(M_9) \oplus K_{10} \quad (4.82)$$

$$D = \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M_9) \oplus K_{10}] \oplus 00 \quad (4.83)$$

By reversing `ShiftRows` and `SubBytes` operations in equation 4.83, we get equation 4.84:

$$\text{SB}^{-1} \circ \text{SR}^{-1}(D) = \text{MC} \circ \text{SR} \circ \text{SB}(M_9) \oplus K_{10} \quad (4.84)$$

Then, an xor operation between equation 4.84 and equation 4.82 gives the new equation 4.85:

$$C \oplus \text{SB}^{-1} \circ \text{SR}^{-1}(D) = \text{SR} \circ \text{SB}(M_9) \oplus \text{MC} \circ \text{SR} \circ \text{SB}(M_9) \quad (4.85)$$

We define X value by equation 4.86 for greater simplicity in writing the equations:

$$X = \text{SR} \circ \text{SB}(M_9) \quad (4.86)$$

We replace X in equation 4.85 and get the new equation 4.87:

$$C \oplus \text{SB}^{-1} \circ \text{SR}^{-1}(D) = X \oplus \text{MC}(X) \quad (4.87)$$

Then, we perform an exhaustive search on values of X in equation 4.87. As there is a **MixColumns** transformation in equation 4.87, the value of each byte of X depends on the other values in the same column. So, we must perform an exhaustive search for the quadruple of bytes in the first column. Then, we repeat the search for each of the next three columns.

Each column hypothesis on the X values gives an hypothesis about the corresponding column of K_{10} using equation 4.88. This exhaustive search among $(2^8)^4$ values for each column of X leads to a set of 2^8 hypotheses for each column and consequently for the corresponding column of K_{10} .

$$K_{10} = C \oplus X \quad (4.88)$$

In the next step, a second exhaustive search is required among all the combinations of column hypotheses about K_{10} . To perform this new search, for each combination of hypotheses about 4 columns of K_{10} , all the previous round keys must be calculated by the reverse of **KeyExpansion** operations. Then, we must encrypt one of the plaintext M in order to examine the validity of the current key hypothesis. As soon as we find C as the result of encryption, the key is revealed and the exhaustive search stops.

Therefore, for creating key column hypotheses, $(2^8)^4 \times 4$ values are required. Then a maximum of $(2^8)^4$ verifications is necessary in order to examine all the key column hypotheses in the second step. However, on average the key is revealed after $(2^8)^4$ searches. So, a first exhaustive search of 2^{34} values, followed by a second search of 2^{31} on average is required for finding the key.

• Attack II-1 without memory access

To exploit this attack without memory access, three pairs of corresponding correct and faulty ciphertexts are required:

$$C^a = \text{SR} \circ \text{SB}(M_9^a) \oplus K_{10} \quad (4.89a)$$

$$C^b = \text{SR} \circ \text{SB}(M_9^b) \oplus K_{10} \quad (4.89b)$$

$$C^c = \text{SR} \circ \text{SB}(M_9^c) \oplus K_{10} \quad (4.89c)$$

$$D^a = \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M_9^a) \oplus K_{10}] \oplus K'_{f=11} \quad (4.90a)$$

$$D^b = \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M_9^b) \oplus K_{10}] \oplus K'_{f=11} \quad (4.90b)$$

$$D^c = \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M_9^c) \oplus K_{10}] \oplus K'_{f=11} \quad (4.90c)$$

Xor operations between the first faulty ciphertext equation and each of the following ones in equations 4.90 give the new equations 4.91:

$$\begin{aligned} D^a \oplus D^b &= \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M_9^a) \oplus K_{10}] \oplus \\ &\quad \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M_9^b) \oplus K_{10}] \end{aligned} \quad (4.91a)$$

$$\begin{aligned} D^a \oplus D^c &= \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M_9^a) \oplus K_{10}] \oplus \\ &\quad \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M_9^c) \oplus K_{10}] \end{aligned} \quad (4.91b)$$

We reverse **ShiftRows** operations in equations 4.91. Then, we replace corresponding values of C^a , C^b and C^c and we obtain equations 4.92:

$$\text{SR}^{-1}(D^a \oplus D^b) = \text{SB}[\text{MC}(C^a \oplus K_{10}) \oplus K_{10}] \oplus \text{SB}[\text{MC}(C^b \oplus K_{10}) \oplus K_{10}] \quad (4.92a)$$

$$\text{SR}^{-1}(D^a \oplus D^c) = \text{SB}[\text{MC}(C^a \oplus K_{10}) \oplus K_{10}] \oplus \text{SB}[\text{MC}(C^c \oplus K_{10}) \oplus K_{10}] \quad (4.92b)$$

According to the distributivity property of **MixColumns**, we transform equations 4.92 to the new equations 4.93:

$$\begin{aligned} \text{SR}^{-1}(D^a \oplus D^b) &= \text{SB}[\text{MC}(C^a) \oplus \text{MC}(K_{10}) \oplus K_{10}] \oplus \\ &\quad \text{SB}[\text{MC}(C^b) \oplus \text{MC}(K_{10}) \oplus K_{10}] \end{aligned} \quad (4.93a)$$

$$\begin{aligned} \text{SR}^{-1}(D^a \oplus D^c) &= \text{SB}[\text{MC}(C^a) \oplus \text{MC}(K_{10}) \oplus K_{10}] \oplus \\ &\quad \text{SB}[\text{MC}(C^c) \oplus \text{MC}(K_{10}) \oplus K_{10}] \end{aligned} \quad (4.93b)$$

C^a , C^b , C^c , D^a , D^b and D^c have known values.

We replace C^a , C^b , C^c , D^a , D^b and D^c by their corresponding values in equations 4.93. Then, we examine all the possible values for K_{10} and **MixColumns** K_{10} on a single column by an exhaustive search in equations 4.93.

This exhaustive search of $(2^8)^4$ values for each column of K_{10} leads to a set of 2^8 hypotheses for the corresponding column values. Repeating these operations for the next three columns creates three further sets of 2^8 hypotheses of column values for the subsequent columns.

Then, a second exhaustive search between column hypotheses reveals a unique value for K_{10} . It requires a procedure for verification of all the column hypotheses on K_{10} . This verification procedure is similar to the second exhaustive search for this attack with memory access. So, for each combination of 4 columns hypotheses on K_{10} , all the previous round keys must be calculated by the reverse of **KeyExpansion** operations. Then, we must encrypt one of the M plaintexts in order to examine the validity of the current key hypothesis. As soon as we find C as the result of encryption, the key value is revealed and the exhaustive search is interrupted.

Consequently, not similar to this attack with memory access, a first exhaustive search of 2^{34} values followed by a second search of 2^{31} on average is required for finding the key. The main difference between the two cases, having access or not having access to memory, is the number of required pairs of corresponding faulty and correct ciphertexts needed to complete this attack.

Scenario II - Attack 2

Attack moment: Any time between $RC=0$ and $RC=10$ FOR

Required fault model: $\{04\}$

Algorithm execution: $R_0..R_9-4 \times R_{m=10..13}-R_{f=14}$

• Attack II-2 with memory access

This attack adds 4 supplementary rounds to normal AES execution. For greater ease, we assume **MiddleRound** and **FinalRound** as two functions that combine different operations of a middle or of a final round according to equations 4.78 and 4.79. In order to exploit this attack, only one pair of corresponding correct and faulty ciphertexts is required. Using the new functions, we write them as 4.94 and 4.95:

$$C = \text{FR}(M_9) \oplus K_{10} \quad (4.94)$$

$$D = \text{FR}(\text{MR}(\text{MR}(\text{MR}(\text{MR}(M_9) \oplus K_{10}) \oplus 00) \oplus 00) \oplus 00) \oplus 00 \quad (4.95)$$

We remove useless operations in equation 4.95 and we rewrite it as equation 4.96:

$$\begin{aligned} D &= \text{FR}(\text{MR}(\text{MR}(\text{MR}(\text{MR}(M_9) \oplus K_{10})))) \\ &= \text{FR} \circ \text{MR} \circ \text{MR} \circ \text{MR}[\text{MR}(M_9) \oplus K_{10}] \end{aligned} \quad (4.96)$$

Then, by reversing **MiddleRound** and **FinalRound** operations in equation 4.96, we obtain equation 4.97:

$$\text{MR}^{-1} \circ \text{MR}^{-1} \circ \text{MR}^{-1} \circ \text{FR}^{-1}(D) = \text{MR}(M_9) \oplus K_{10} \quad (4.97)$$

We rewrite the `MiddleRound` operation on the right side of equation 4.97 as `MixColumns` $\circ$ `FinalRound` to get closer to equation 4.94. Thus, we obtain equation 4.98:

$$\text{MR}^{-1} \circ \text{MR}^{-1} \circ \text{MR}^{-1} \circ \text{FR}^{-1}(D) = \text{MC} \circ \text{FR}(M_9) \oplus K_{10} \quad (4.98)$$

We enter a new variable X which is defined in equation 4.99. For greater ease, we replace $\text{FR}(M_9)$ values by X in equation 4.98:

$$X = \text{FR}(M_9) \quad (4.99)$$

Thus, we obtain equation 4.100:

$$\text{MR}^{-1} \circ \text{MR}^{-1} \circ \text{MR}^{-1} \circ \text{FR}^{-1}(D) = \text{MC}(X) \oplus K_{10} \quad (4.100)$$

Then, we perform an xor operation between equations 4.94 and 4.100. We obtain equation 4.101:

$$C \oplus \text{MR}^{-1} \circ \text{MR}^{-1} \circ \text{MR}^{-1} \circ \text{FR}^{-1}(D) = X \oplus \text{MC}(X) \quad (4.101)$$

C and D have known values.

Now, we perform an exhaustive search of the first column of X values, according to equation 4.101. The exhaustive search leads to a set of hypotheses on the corresponding column of K_{10} . We repeat these searches for the following columns of X and K_{10} .

Afterwards, we perform a second exhaustive search between K_{10} columns hypotheses. For each combination of four columns hypotheses, we calculate all the round keys in order to examine these current column hypotheses.

This two-step exhaustive search and its complexity are similar to the one described exploitation in attack 1 of scenario II.

• Attack II-2 without memory access

A pair of corresponding correct and faulty ciphertexts in this attack can be written as equations 4.102 and 4.103:

$$C = \text{FR}(M_9) \oplus K_{10} \quad (4.102)$$

$$D = \text{FR}(\text{MR}(\text{MR}(\text{MR}(\text{MR}(M_9) \oplus K_{10}) \oplus K'_{11}) \oplus K'_{12}) \oplus K'_{13}) \oplus K'_{14} \quad (4.103)$$

$K'_{11}, K'_{12}, K'_{13}, K'_{14}$ are invalid keys with unknown values.

Exploitation of this attack needs exhaustive searches for the values of four erroneous additional rounds. The needed calculations cannot be done in a reasonable amount of time. Even, by using several pairs of corresponding correct and faulty ciphertexts, we cannot adequately reduce the calculation time.

Scenario II - Attack 3

Attack moment: Any time between $RC=8$ LOOP and $RC=9$ FOR

Required fault model: $\{02, 08\}$

Algorithm execution: $R_0..R_8-R_{f=9}$

• Attack II-3 with or without memory access

For exploiting this attack, three pairs of corresponding faulty and correct ciphertexts are required:

$$C^a = \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M_8^a) \oplus K_9] \oplus K_{10} \quad (4.104a)$$

$$C^b = \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M_8^b) \oplus K_9] \oplus K_{10} \quad (4.104b)$$

$$C^c = \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M_8^c) \oplus K_9] \oplus K_{10} \quad (4.104c)$$

$$D^a = \text{SR} \circ \text{SB}(M_8^a) \oplus K_{f=9} \quad (4.105a)$$

$$D^b = \text{SR} \circ \text{SB}(M_8^b) \oplus K_{f=9} \quad (4.105b)$$

$$D^c = \text{SR} \circ \text{SB}(M_8^c) \oplus K_{f=9} \quad (4.105c)$$

If we perform an xor operation between the first equation 4.104 and each of the following ones, we obtain the equations 4.106:

$$C^a \oplus C^b = \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M_8^a) \oplus K_9] \oplus \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M_8^b) \oplus K_9] \quad (4.106a)$$

$$C^a \oplus C^c = \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M_8^a) \oplus K_9] \oplus \text{SR} \circ \text{SB}[\text{MC} \circ \text{SR} \circ \text{SB}(M_8^c) \oplus K_9] \quad (4.106b)$$

By using the reverse **ShiftRows** function in equation 4.106 and by replacing D^a , D^b and D^c according to their values in equations 4.105, we get new equations 4.107:

$$\text{SR}^{-1}(C^a \oplus C^b) = \text{SB}[\text{MC}(D^a \oplus K_9) \oplus K_9] \oplus \text{SB}[\text{MC}(D^b \oplus K_9) \oplus K_9] \quad (4.107a)$$

$$\text{SR}^{-1}(C^a \oplus C^c) = \text{SB}[\text{MC}(D^a \oplus K_9) \oplus K_9] \oplus \text{SB}[\text{MC}(D^c \oplus K_9) \oplus K_9] \quad (4.107b)$$

According to the distributivity property of `MixColumns`, we transform equations 4.107 to equations 4.108:

$$\text{SR}^{-1}(C^a \oplus C^b) = \text{SB}[\text{MC}(D^a) \oplus \text{MC}(K_9) \oplus K_9] \oplus \text{SB}[\text{MC}(D^b) \oplus \text{MC}(K_9) \oplus K_9] \quad (4.108a)$$

$$\text{SR}^{-1}(C^a \oplus C^c) = \text{SB}[\text{MC}(D^a) \oplus \text{MC}(K_9) \oplus K_9] \oplus \text{SB}[\text{MC}(D^c) \oplus \text{MC}(K_9) \oplus K_9] \quad (4.108b)$$

C^a, C^b, C^c, D^a, D^b and D^c have known values.

We replace C^a, C^b, C^c, D^a, D^b and D^c in equations 4.108 by their corresponding values. Then, we examine all the possible values for K_9 and `MixColumns` (K_9) by an exhaustive search of their first columns in equations 4.108.

This exhaustive search among $(2^8)^4$ values for each column of K_9 leads to a set 2^8 hypotheses for the column value. By repeating this procedure for each of three following columns, we get a new set of 2^8 hypotheses for each column.

Then, another exhaustive search is required among all the combinations of hypotheses in order to find the unique key value. At this stage, for each combination of column hypotheses for K_9 , all the previous and following round keys must be calculated. Then, the candidate keys must be applied to ciphering one of the plaintexts, e.g. M^a , in order to determine if the encryption results in C^a . As soon as the first key hypothesis is detected as the unique corresponding key, the exhaustive search stops.

Therefore, the second exhaustive search is performed on $(2^8)^4$ hypotheses of key columns; but the key is normally revealed after an average of $(2^8)^3$ calculations.

Consequently, 4 exhaustive searches of $(2^8)^4$ values for key columns, (equal to 2^{32} ,) lead to a new exhaustive search of 2^{31} values.

This attack is independent from any invalid key value. Consequently, there is no difference in this exploitation whether the attack is with or without memory access.

4.2.3 Secondary Attack Scenarios

In addition to the main attack scenarios described in the previous section, two other scenarios are feasible. Each of these scenarios includes two attacks targeting two consecutive encryptions of the same plaintext. We present them in the sections 4.2.3.1 and 4.2.3.2.

4.2.3.1 Double Attack on the Round Counter Value (Scenario III)

This scenario is based on differential analysis of two consecutive faulty encryptions of the same plaintext. In both attacks, the round counter is targeted, during the execution of similar algorithm stages of two consecutive rounds. Therefore, by using this scenario, a longer possible time for the attack (during AES execution) is provided

to the opponent. The double attack scenario requires two round-reduced encryptions due to similar attacks on two consecutive rounds.

We show a summary of all exploitable attacks with single-bit faults in table [4.6](#):

Table 4.6: The exploitable attacks of Scenario III.

Round counter	Attack moment	Fault model							
		0x01	0x02	0x04	0x08	0x10	0x20	0x40	0x80
$RC = 1$		$RC \rightarrow 0$	$RC \rightarrow 3$	$RC \rightarrow 5$	$RC \rightarrow 9$	$RC \rightarrow 17$	$RC \rightarrow 33$	$RC \rightarrow 65$	$RC \rightarrow 129$
	FOR					$NR \rightarrow 1$ $R_0 \cdot R_f$ $f=17$	$NR \rightarrow 1$ $R_0 \cdot R_f$ $f=33$	$NR \rightarrow 1$ $R_0 \cdot R_f$ $f=65$	$NR \rightarrow 1$ $R_0 \cdot R_f$ $f=129$
	LOOP					$NR \rightarrow 2$ $R_0 \cdot R_m \cdot R_f$ $m=17$ $f=18$	$NR \rightarrow 2$ $R_0 \cdot R_m \cdot R_f$ $m=33$ $f=34$	$NR \rightarrow 2$ $R_0 \cdot R_m \cdot R_f$ $m=65$ $f=66$	$NR \rightarrow 2$ $R_0 \cdot R_m \cdot R_f$ $m=129$ $f=130$
	ARK COMEBACK					$NR \rightarrow 2$ $R_0 \cdot R_1 \cdot R_f$ $f=18$	$NR \rightarrow 2$ $R_0 \cdot R_1 \cdot R_f$ $f=34$	$NR \rightarrow 2$ $R_0 \cdot R_1 \cdot R_f$ $f=66$	$NR \rightarrow 2$ $R_0 \cdot R_1 \cdot R_f$ $f=130$
$RC = 2$		$RC \rightarrow 3$	$RC \rightarrow 0$	$RC \rightarrow 6$	$RC \rightarrow 10$	$RC \rightarrow 18$	$RC \rightarrow 34$	$RC \rightarrow 66$	$RC \rightarrow 130$
	FOR					$NR \rightarrow 2$ $R_0 \cdot R_1 \cdot R_f$ $f=18$	$NR \rightarrow 2$ $R_0 \cdot R_1 \cdot R_f$ $f=34$	$NR \rightarrow 2$ $R_0 \cdot R_1 \cdot R_f$ $f=66$	$NR \rightarrow 2$ $R_0 \cdot R_1 \cdot R_f$ $f=130$
	LOOP					$NR \rightarrow 3$ $R_0 \cdot R_1 \cdot R_m \cdot R_f$ $m=18$ $f=19$	$NR \rightarrow 3$ $R_0 \cdot R_1 \cdot R_m \cdot R_f$ $m=34$ $f=35$	$NR \rightarrow 3$ $R_0 \cdot R_1 \cdot R_m \cdot R_f$ $m=66$ $f=67$	$NR \rightarrow 3$ $R_0 \cdot R_1 \cdot R_m \cdot R_f$ $m=130$ $f=131$
	ARK COMEBACK				$NR \rightarrow 3$ $R_0 \cdot R_2 \cdot R_f$ $f=11$	$NR \rightarrow 3$ $R_0 \cdot R_2 \cdot R_f$ $f=19$	$NR \rightarrow 3$ $R_0 \cdot R_2 \cdot R_f$ $f=35$	$NR \rightarrow 3$ $R_0 \cdot R_2 \cdot R_f$ $f=67$	$NR \rightarrow 3$ $R_0 \cdot R_2 \cdot R_f$ $f=131$

Table continued on next page

Table 4.6 – Table continued from previous page

Round counter	Attack moment	Fault model							
		0x01	0x02	0x04	0x08	0x10	0x20	0x40	0x80
$RC = 5$		$RC \rightarrow 4$	$RC \rightarrow 7$	$RC \rightarrow 1$	$RC \rightarrow 13$	$RC \rightarrow 21$	$RC \rightarrow 37$	$RC \rightarrow 69$	$RC \rightarrow 133$
	FOR				$NR \rightarrow 5$ $R_0 \dots R_4 \text{-} R_f$ $f=13$	$NR \rightarrow 5$ $R_0 \dots R_4 \text{-} R_f$ $f=21$	$NR \rightarrow 5$ $R_0 \dots R_4 \text{-} R_f$ $f=37$	$NR \rightarrow 5$ $R_0 \dots R_4 \text{-} R_f$ $f=69$	$NR \rightarrow 5$ $R_0 \dots R_4 \text{-} R_f$ $f=133$
	LOOP				$NR \rightarrow 6$ $R_0 \dots R_4 \text{-} R_m \text{-} R_f$ $m=13$ $f=14$	$NR \rightarrow 6$ $R_0 \dots R_4 \text{-} R_m \text{-} R_f$ $m=21$ $f=22$	$NR \rightarrow 6$ $R_0 \dots R_4 \text{-} R_m \text{-} R_f$ $m=37$ $f=38$	$NR \rightarrow 6$ $R_0 \dots R_4 \text{-} R_m \text{-} R_f$ $m=69$ $f=70$	$NR \rightarrow 6$ $R_0 \dots R_4 \text{-} R_m \text{-} R_f$ $m=133$ $f=134$
	ARK COMEBACK				$NR \rightarrow 6$ $R_0 \dots R_5 \text{-} R_f$ $f=14$	$NR \rightarrow 6$ $R_0 \dots R_5 \text{-} R_f$ $f=22$	$NR \rightarrow 6$ $R_0 \dots R_5 \text{-} R_f$ $f=38$	$NR \rightarrow 6$ $R_0 \dots R_5 \text{-} R_f$ $f=70$	$NR \rightarrow 6$ $R_0 \dots R_5 \text{-} R_f$ $f=134$
$RC = 6$		$RC \rightarrow 7$	$RC \rightarrow 4$	$RC \rightarrow 2$	$RC \rightarrow 14$	$RC \rightarrow 22$	$RC \rightarrow 38$	$RC \rightarrow 70$	$RC \rightarrow 134$
	FOR				$NR \rightarrow 6$ $R_0 \dots R_5 \text{-} R_f$ $f=14$	$NR \rightarrow 6$ $R_0 \dots R_5 \text{-} R_f$ $f=22$	$NR \rightarrow 6$ $R_0 \dots R_5 \text{-} R_f$ $f=38$	$NR \rightarrow 6$ $R_0 \dots R_5 \text{-} R_f$ $f=70$	$NR \rightarrow 6$ $R_0 \dots R_5 \text{-} R_f$ $f=134$
	LOOP				$NR \rightarrow 7$ $R_0 \dots R_5 \text{-} R_m \text{-} R_f$ $m=14$ $f=15$	$NR \rightarrow 7$ $R_0 \dots R_5 \text{-} R_m \text{-} R_f$ $m=22$ $f=23$	$NR \rightarrow 7$ $R_0 \dots R_5 \text{-} R_m \text{-} R_f$ $m=38$ $f=39$	$NR \rightarrow 7$ $R_0 \dots R_5 \text{-} R_m \text{-} R_f$ $m=70$ $f=71$	$NR \rightarrow 7$ $R_0 \dots R_5 \text{-} R_m \text{-} R_f$ $m=134$ $f=135$
	ARK COMEBACK				$NR \rightarrow 7$ $R_0 \dots R_6 \text{-} R_f$ $f=15$	$NR \rightarrow 7$ $R_0 \dots R_6 \text{-} R_f$ $f=23$	$NR \rightarrow 7$ $R_0 \dots R_6 \text{-} R_f$ $f=39$	$NR \rightarrow 7$ $R_0 \dots R_6 \text{-} R_f$ $f=71$	$NR \rightarrow 7$ $R_0 \dots R_6 \text{-} R_f$ $f=135$

Table continued on next page

Table 4.6 – Table continued from previous page

Round counter	Attack moment	Fault model							
		0x01	0x02	0x04	0x08	0x10	0x20	0x40	0x80
<i>RC</i> = 9		<i>RC</i>→8	<i>RC</i>→11	<i>RC</i>→13	<i>RC</i>→1	<i>RC</i>→25	<i>RC</i>→41	<i>RC</i>→73	<i>RC</i>→137
	FOR		<i>NR</i> →9 <i>R</i> ₀ .. <i>R</i> ₈ - <i>R</i> _f <i>f</i> =11	<i>NR</i> →9 <i>R</i> ₀ .. <i>R</i> ₈ - <i>R</i> _f <i>f</i> =13		<i>NR</i> →9 <i>R</i> ₀ .. <i>R</i> ₈ - <i>R</i> _f <i>f</i> =25	<i>NR</i> →9 <i>R</i> ₀ .. <i>R</i> ₈ - <i>R</i> _f <i>f</i> =41	<i>NR</i> →9 <i>R</i> ₀ .. <i>R</i> ₈ - <i>R</i> _f <i>f</i> =73	<i>NR</i> →9 <i>R</i> ₀ .. <i>R</i> ₈ - <i>R</i> _f <i>f</i> =137
	LOOP		<i>NR</i> →10 <i>R</i> ₀ .. <i>R</i> ₈ - <i>R</i> _m - <i>R</i> _f <i>m</i> =11 <i>f</i> =12	<i>NR</i> →10 <i>R</i> ₀ .. <i>R</i> ₈ - <i>R</i> _m - <i>R</i> _f <i>m</i> =13 <i>f</i> =14		<i>NR</i> →10 <i>R</i> ₀ .. <i>R</i> ₈ - <i>R</i> _m - <i>R</i> _f <i>m</i> =25 <i>f</i> =26	<i>NR</i> →10 <i>R</i> ₀ .. <i>R</i> ₈ - <i>R</i> _m - <i>R</i> _f <i>m</i> =41 <i>f</i> =42	<i>NR</i> →10 <i>R</i> ₀ .. <i>R</i> ₈ - <i>R</i> _m - <i>R</i> _f <i>m</i> =73 <i>f</i> =74	<i>NR</i> →10 <i>R</i> ₀ .. <i>R</i> ₈ - <i>R</i> _m - <i>R</i> _f <i>m</i> =137 <i>f</i> =138
	ARK COMEBACK		<i>NR</i> →10 <i>R</i> ₀ .. <i>R</i> ₉ - <i>R</i> _f <i>f</i> =12	<i>NR</i> →10 <i>R</i> ₀ .. <i>R</i> ₉ - <i>R</i> _f <i>f</i> =14		<i>NR</i> →10 <i>R</i> ₀ .. <i>R</i> ₉ - <i>R</i> _f <i>f</i> =26	<i>NR</i> →10 <i>R</i> ₀ .. <i>R</i> ₉ - <i>R</i> _f <i>f</i> =42	<i>NR</i> →10 <i>R</i> ₀ .. <i>R</i> ₉ - <i>R</i> _f <i>f</i> =74	<i>NR</i> →10 <i>R</i> ₀ .. <i>R</i> ₉ - <i>R</i> _f <i>f</i> =138
<i>RC</i> = 10		<i>RC</i>→11	<i>RC</i>→8	<i>RC</i>→14	<i>RC</i>→2	<i>RC</i>→26	<i>RC</i>→42	<i>RC</i>→74	<i>RC</i>→138
	FOR	<i>NR</i> →10 <i>R</i> ₀ .. <i>R</i> ₉ - <i>R</i> _f <i>f</i> =11		<i>NR</i> →10 <i>R</i> ₀ .. <i>R</i> ₉ - <i>R</i> _f <i>f</i> =14		<i>NR</i> →10 <i>R</i> ₀ .. <i>R</i> ₉ - <i>R</i> _f <i>f</i> =26	<i>NR</i> →10 <i>R</i> ₀ .. <i>R</i> ₉ - <i>R</i> _f <i>f</i> =42	<i>NR</i> →10 <i>R</i> ₀ .. <i>R</i> ₉ - <i>R</i> _f <i>f</i> =74	<i>NR</i> →10 <i>R</i> ₀ .. <i>R</i> ₉ - <i>R</i> _f <i>f</i> =138
	LOOP								

4.2.3.2 Double Attack on the Round Counter Value and the Round Counter Reference (Scenario IV)

The second scenario of double attacks is based on two consecutive fault injections. The first attack must be done on the round counter (RC) followed by the second attack targeting the total round number reference (R_{max}).

4.2.4 Cryptanalysis of Secondary Attacks

4.2.4.1 Cryptanalytic Solutions For Exploitable Attacks of Scenario III

The 1st attack can be realized in one of the two following moments:

1st moment option: $RC=n$ ARK, COMEBACK or $RC=n+1$ FOR
 $2 \leq n \leq 8$; also feasible for $n=1$ and $n=9$, but this option is of no interest.
 Required fault model: $\{10, 20, 40, 80\} \cup \{\text{other corresponding faults as mentioned in appendix A}\}$
 Algorithm execution: $R_0..R_n-R_{f1}$

or

2nd moment option: $RC=n$ LOOP
 $2 \leq n \leq 8$; also feasible for $n=1$ and $n=9$, but this option is of no interest.
 Required fault model: $\{10, 20, 40, 80\} \cup \{\text{other corresponding faults as mentioned in appendix A}\}$
 Algorithm execution: $R_0..R_{n-1}-R_{m1}-R_{f1}$

For the second attack, two different moments can be also considered:

1st moment option: $RC=n+1$ LOOP
 Required fault model: $\{10, 20, 40, 80\} \cup \{\text{other corresponding faults as mentioned in appendix A}\}$
 Algorithm execution: $R_0..R_n-R_{m2}-R_{f2}$

or

2nd moment option: $RC=n+1$ ARK, COMEBACK or $RC=n+2$ FOR
 Required fault model: $\{10, 20, 40, 80\} \cup \{\text{other corresponding faults as mentioned in appendix A}\}$
 Algorithm execution: $R_0..R_{n+1}-R_{f2}$

Therefore, each of the four possible combinations of a 1st and a 2nd attacks leads to a special cryptanalytic solution exploiting the differential round values.

4.3 Conclusions

In this chapter, we extended our research to explore new models of fault attacks. Fault injection attacks for Round Modification Analysis have been studied much less than those for DFA methods. With the precision obtained in previous chapters,

we developed a couple of new attacks with their associated cryptanalytic solutions. They have two targets: the round counter and the reference number of AES rounds. Eleven attacks are based on single-target scenarios, including an attack at a specific time of algorithm execution, on one of the two aforementioned targets. Our experiments showed the large possibility of Round Modification Analysis by fault injection.

Therefore, the implementation of appropriate countermeasures is necessary for these kinds of attacks targeting the execution of an iterative cryptographic algorithm.

Countermeasures

Contents

5.1	Introduction	119
5.2	Countermeasures against Differential Fault Analysis	120
5.2.1	A Countermeasure against Fault Attacks on the Round Keys	120
5.2.2	A Countermeasure against Attacks on the KeyExpansion	122
5.2.3	A Combined Countermeasure against DFA Attacks	123
5.2.4	Improvement of the Combined Countermeasure against DFA Attacks	124
5.3	Countermeasures Against Round Modification Attacks	127
5.3.1	An Unrolled AES Countermeasure	127
5.3.2	A More Secure Unrolled AES Countermeasure	128
5.4	A Combined Countermeasure against DFA and RMA	130
5.5	Conclusions	130

5.1 Introduction

In chapters 3 and 4, we reported two attack schemes on cryptosystems by laser fault injection. These attacks may menace security of circuit contents and provide opponents with access to secrets. In this chapter, we present several protective solutions that may be used as *countermeasures* against these threats.

According to [Vacca 2009] and [CNSS 2010], countermeasures or *security safeguards* refer to protective measures and controls for system security requirements. A countermeasure may be an action, device, procedure, or technique in hardware and software security features, but it is not necessarily limited to these areas. Personnel security as well as physical structures, areas, and devices may be intended to safeguard elements of a cryptographic system.

The objective of countermeasures is to reduce the feasibility of an attack by eliminating or preventing it, by minimizing the damage it can produce, or by detecting and reporting it, so that proper actions can be taken [CNSS 2010].

In the coming sections of this chapter, we present a set of countermeasures against the DFA and RMA attacks described in the previous chapters.

5.2 Countermeasures against Differential Fault Analysis

In chapter 3, we described how an opponent may discover a secret key by laser fault injection and DFA processes. In DFA attacks on AES, required fault models are usually single or several faulty bits or bytes injected at a certain stage of the AES execution. Attacks on corresponding round keys are often alternative solutions for generating required faults on the temporary ciphertext. In our experiments, described in chapter 3, we targeted round keys as a more practical solution with a longer time constraint in comparison to injecting faults directly into the chip's ALU. We generated the required faults on the temporary ciphertexts via faulty round keys. Therefore, in this section we study a solution which prevents fault injection on round keys.

For protecting our circuit, the ideal countermeasure must protect the round keys from any fault injection by laser. It would also be very appreciable if this countermeasure safeguarded the circuit against other means of fault attacks. However, this desired countermeasure depends on layout design issues and research. So, it is out of this thesis' scope. In this chapter we only survey embedded techniques and protections against these kinds of attacks. Besides, this presumptive hardware solution may be vulnerable to future etching methods or newer surgical laser fault injection techniques. To this end, even when using hardware countermeasures, the embedded protections are still strongly needed.

A well designed embedded countermeasure against fault attacks on the round keys must immediately detect any change in their values and must prevent the release of faulty ciphertext. Therefore, faulty results containing secret information will be hidden from the opponent and there will remain no possibility for DFA.

5.2.1 A Countermeasure against Fault Attacks on the Round Keys

In section 2.2 we described that in our AES implementation all the round keys were derived only once after circuit reset and stored in SRAM. Then, for each text encryption, the `AddRoundKey` operation refers to stored round keys in SRAM without recomputing them.

In our circuit, the stored round keys in SRAM are potential targets for fault attacks. In our experiments, reported in chapter 3, we illustrated that effective DFA attacks are feasible by laser fault injection on round keys. We described the danger of such attacks on the cryptosystems, even when the number of injected faults is too big and does not fit the model. Therefore, protective measures are required for guaranteeing the security of cryptosystems.

For preventing the use of faulty round keys, we added two security transformations:

- `Kxoring` operation to `KeyExpansion` processes
- `KeyTest` operation to `DataEncryption` processes

Kxoring is based on information redundancy which adds bitwise parity to the columns. It computes $KxorRef$ for each derived round key during **KeyExpansion** processes. Then, in **DataEncryption** processes, **KeyTest** calculates $Kxor$ value for the last used round key after each **AddRoundKey**. **KeyTest** checks the integrity of the used round key by comparing current $Kxor$ to corresponding $KxorRef$.

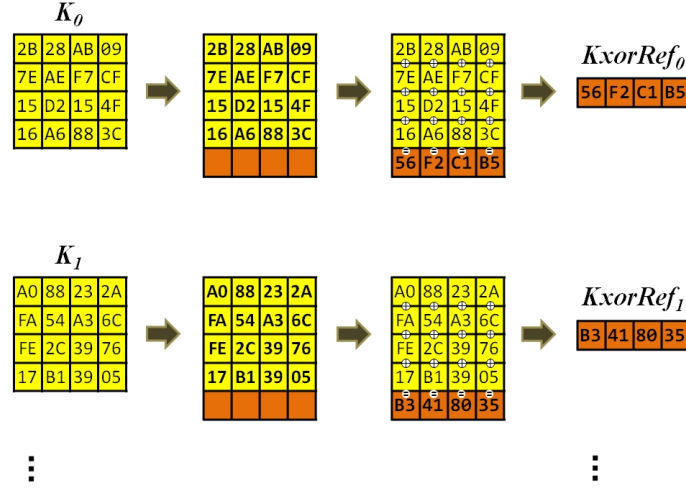


Figure 5.1: Calculation of $KxorRef$ bytes.

$KxorRef$ is an array of 11 blocs, similar to the number of round keys. Each bloc is composed of 4 bytes. Each byte stores the xor product of corresponding sub-key column cells. Figure 5.1 shows how the $KxorRef$ bytes are calculated. The computations are done during **KeyExpansion** in parallel to the standard **KeyScheduling** operations. For instance after **KeyScheduling** for K_i values, $KxorRef_i$ is calculated by the following operations:

$$\begin{aligned}
 KxorRef[i][0] &\leftarrow K[i][0] \oplus K[i][1] \oplus K[i][2] \oplus K[i][3] \\
 KxorRef[i][1] &\leftarrow K[i][4] \oplus K[i][5] \oplus K[i][6] \oplus K[i][7] \\
 KxorRef[i][2] &\leftarrow K[i][8] \oplus K[i][9] \oplus K[i][10] \oplus K[i][11] \\
 KxorRef[i][3] &\leftarrow K[i][12] \oplus K[i][13] \oplus K[i][14] \oplus K[i][15]
 \end{aligned}$$

$KxorRef$ values are used during **DataEncryption** for checking the integrity of used round keys. $Kxor$ is a variable, composed of a bloc of 4 bytes. It is used for the checking operation during **DataEncryption**.

Table 5.1 shows K_{RC} and $KxorRef_{RC}$ values for a key equal to 2B7E151628AED2A6ABF7158809CF4F3C.

During each encryption process, $Kxor$ is used for checking the integrity of stored round key values. At the end of each **AddRoundKey** operation, $Kxor$ is calculated. Then **KeyTest** operation checks if $Kxor$ is equal to the current round's $KxorRef$ value, as shown in the instructions below:

$$C \leftarrow C \oplus K[RC]$$

```

IF (Kxor(K[RC]) != KxorRef[RC]) THEN
  C ← 0
  BREAK
ENDIF

```

Any difference indicates the existence of a fault on the current round key. Therefore, the countermeasure takes appropriate action and returns a ciphertext containing only zero, in order to neutralize any DFA threat.

Table 5.1: An example for K_{RC} and $KxorRef_{RC}$ values.

RC	$K[RC]$	$KxorRef[RC]$
0	2B7E151628AED2A6ABF7158809CF4F3C	56F2C1B5
1	A0FAFE1788542CB123A339392A6C7605	B3418035
2	F2C295F27A96B9435935807A7359F67F	571696A3
3	3D80477D4716FE3E1E237E446D7A883B	879107A4
4	EF44A541A8525B7FB671253BDB0BAD00	4FDED97D
5	D4D1C6F87C839D87CAF2B8BC11F915BC	3BE53C41
6	6D88A37A110B3EFDDBF98641CA0093FD	3CD9E5A4
7	4E54F70E5F5FC9F384A64FB24EA6DC4F	E33ADF7B
8	EAD27321B58DBAD2312BF5607F8D292F	6A508FF4
9	AC7766F319FADC2128D12941575C006E	4E1E9165
10	D014F9A8C9EE2589E13F0CC8B6630CA6	958B1A7F

KeyTest operation is done after **AddRoundKey** and not before. So, there is no chance to inject undetectable faults on the round keys between these two operations.

The duplication of the round keys is another solution for checking their integrity. By using $Kxor$ and $KxorRef$, we reduce the feasibility of reproducing same fault values on duplicated round keys. Figure 5.2 shows how $KxorRef$ calculations and **KeyTest** operations are integrated into **KeyExpansion** and **DataEncryption** processes.

This countermeasure protects the round key values after **KeyExpansion** stage and during all the encryption processes. It is an effective protection against fault injection on stored round keys in SRAM. Nevertheless, it cannot guarantee that the round keys are not faulted during **KeyExpansion**. In order to protect the integrity of round keys during **KeyExpansion**, we need a complementary countermeasure.

5.2.2 A Countermeasure against Attacks on the KeyExpansion

Round keys may also be targeted by fault injection during **KeyExpansion**. Several theoretical DFA models, for instance Giraud’s single-byte [Giraud 2005], are based on injecting fault into a specified round key during the **KeyExpansion** stage.

For this requirement, we need to examine the integrity of stored round keys at the end of **KeyExpansion**, by reversing derivation operations for each stored round key. By examining the **KeyCompare** operation, we determine whether or not reversed **AddRoundKey** operations lead to the secret key. Once this test is

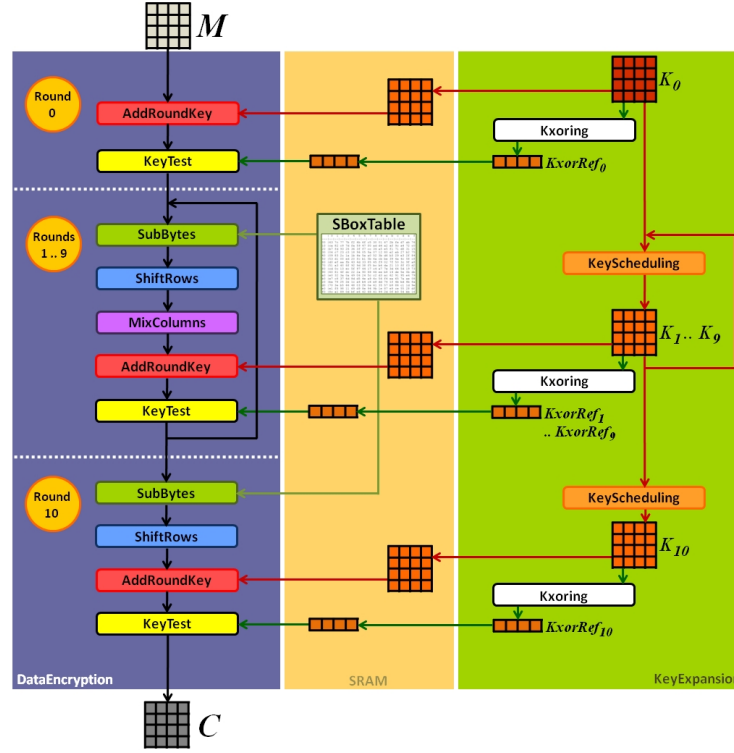


Figure 5.2: Key-protection countermeasure by Kxoring calculation and KeyTest operation in our AES implementation.

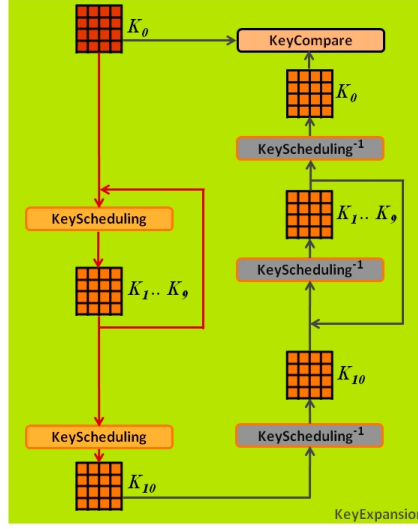
completed, the integrity of stored round keys is controlled by the **KeyTest** countermeasure during the entire encryption processes. Figure 5.3 shows an overview of reversed **KeyScheduling** operations for **KeyCompare** countermeasure in our AES **KeyExpansion** implementation.

Using the reverse of **KeyExpansion** operation for checking the integrity of round keys is a classical countermeasure that has been used before by other researchers. It is done only once at the end of **KeyExpansion** operations. Therefore, it does not increase the **DataEncryption** time.

5.2.3 A Combined Countermeasure against DFA Attacks

The two previous countermeasures protect the round keys from any fault injection. However, they cannot protect the encryption processes against any other possible attacks, such as an eventual fault injection into the chip's ALU or into data buses. Therefore, an additional countermeasure is strongly recommended for safeguarding the encryption processes.

For this reason, we modified our algorithm implementation by adding a secondary and parallel AES encryption process and the **CipherTest** operation at the end of the final round. The secondary encryption is done after end of the first round.

Figure 5.3: **KeyCompare** countermeasure.

Then, both of final ciphertexts are compared to each other using **CipherTest** operation at the end of the second encryption. In case of any difference in the two final ciphertexts is detected, **CipherTest** changes the ciphertext value to 0 and so neutralizes any possibility of DFA by fault injection during the **DataEncryption**.

Besides, the look-up table for **SubBytes** transformations is also stored in SRAM and constitutes another potential target for fault attacks. Hence, we duplicated **SubBytes** look-up table in order to replace value mappings from the secondary encryption path.

We embedded these multiple countermeasures into our circuit. Figure 5.4 shows an overview of parallel encryption processes, **CipherTest**, **KeyTest** and **KeyCompare** operations in our AES implementation.

In our new experiment, we could not inject any fault without its detection by the countermeasures and consequently could not retrieve any of the faulty ciphertexts. These solutions provide the required security against DFA, in the context of our attacks and protect against the release of any faulty ciphertexts.

5.2.4 Improvement of the Combined Countermeasure against DFA Attacks

Our proposed countermeasures concern only the software implementation. They increase the CPU charge, calculation time and memory usage. In the scope of our work, we unfortunately cannot increase the hardware resources or create parallel data flows.

The use of more memory space does not effect the encryption processes, but, the increase of calculation time is a weakness in the performance of our cryptosystem.

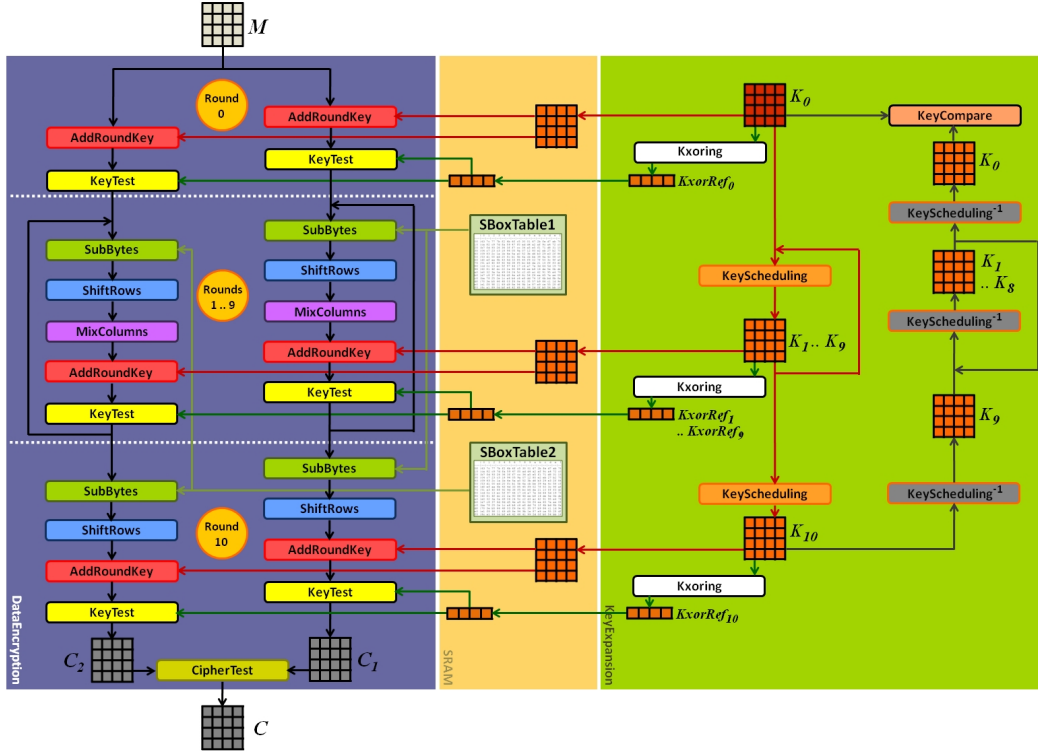


Figure 5.4: Implementation of parallel encryption processes and **KeyTest** and **KeyCompare** operations in our AES implementation.

In the first step, we added **Kxorring** operations and we also added the reverse of **KeyScheduling** to the standard **KeyExpansion**. These operations are done just once and have no effect on the **DataEncryption** processing time.

Our countermeasures increased the **DataEncryption** operations by adding **KeyTest** and by duplicating of all the operations. Consequently, the encryption time of each text is increased by more than 110%. Therefore, we searched for lighter solutions without compromising the security requirements.

According to figure 3.6 and the descriptions in chapter 3 an early fault injected to the temporary ciphertext before **MixColumns** of R_7 is not exploitable by light DFA calculations. Consequently, we need only to protect the temporary ciphertext from faults after **MixColumns** exit of R_7 . For further security against any probable or future DFA attack, we expand the required protected area to an additional round. Thus, we focus on protecting the **DataEncryption** uniquely from **MixColumns** exit of R_6 through the end of R_{10} .

In this case, we can waive **KeyTest** operations between R_0 and R_5 , as well as the parallel **DataEncryption** processes before **MixColumns** entry of R_6 .

Therefore, we duplicate **DataEncryption** only from input of **MixColumns** in R_6 until the end of R_{10} . Thus, if the two ciphertexts are equal, they will be released;

otherwise, they will be set to 0.

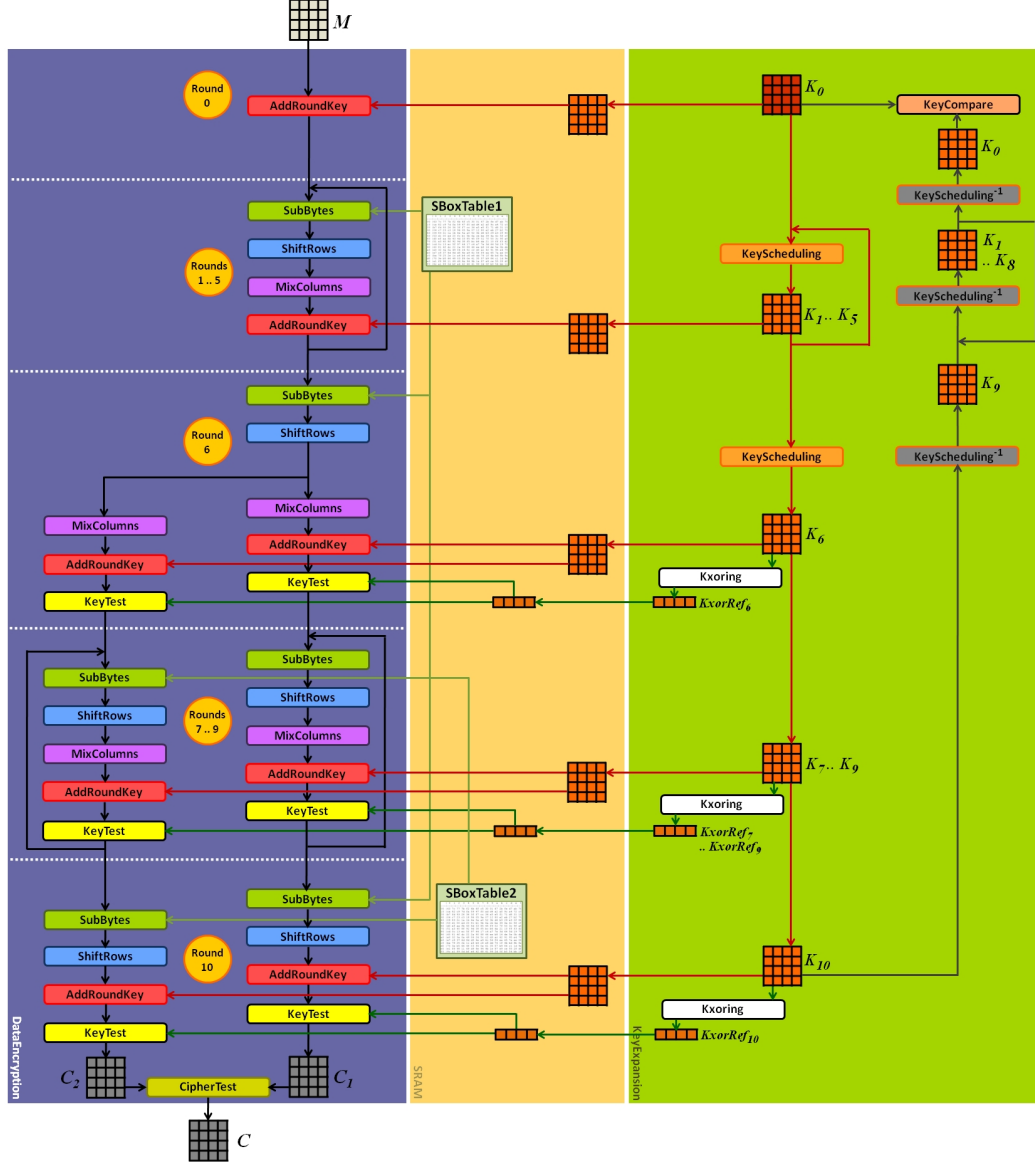


Figure 5.5: Implementation of the improved countermeasures in our AES implementation.

Figure 5.5 shows implementation of our improved countermeasures. In our new experiment they provide the required security against DFA in shorter execution time when compared to our previous combined countermeasures. Besides, they do not implement any protection against round modification attacks. In the next section, we present our countermeasures against RMA on our circuit.

5.3 Countermeasures Against Round Modification Attacks

In chapter 4, we illustrated several algorithm modification attacks by fault round reduction or by fault round addition. Now, in the current section, we present appropriate countermeasures against these kinds of attacks.

5.3.1 An Unrolled AES Countermeasure

Our AES algorithm software implementation is described in chapter 4. Our algorithm has a round counter and a maximum round limit value, called RC and R_{max} respectively, as shown below:

```

START
C ← M
C ← C ⊕ K[0]
FOR RC = 1 TO RC < R_max
    C ← SBOX(C)
    C ← SR(C)
    C ← MC(C)
    C ← C ⊕ K[RC]
    RC ← RC + 1
ENDFOR
C ← SBOX(C)
C ← SR(C)
C ← C ⊕ K[RC]
END

```

During our experiments described in chapter 4, we saw that these two values can be targeted by a laser fault attack. By using appropriate cryptanalysis on the faulty ciphertexts, the opponent may discover the key value.

We need a solution that protects our circuit against any round modification by laser fault injection on RC and R_{max} . We examined the possibility of removing these two potential targets. Therefore, we unrolled AES middle round implementation in order that each round be executed without any loop. The result is an unrolled AES algorithm as shown below:

```

START
C ← M
//ROUND 0
    C ← C ⊕ K[0]
//ROUND 1
    C ← SBOX(C)
    C ← SR(C)
    C ← MC(C)
    C ← C ⊕ K[1]
    ⋮

```

```

//ROUND 10
C ← SBOX(C)
C ← SR(C)
C ← C ⊕ K[10]
END

```

We examined this AES implementation on our circuit. Laser attacks could no longer change neither the round in execution, nor the total number of executed rounds. Nevertheless, it seems that an attack on the *program counter* may cause a jump in the algorithm and which result another round modification attack.

In our experiments, we did not have any success in changing the program counter by laser attack. However, we took into account the possibility of fault attack on the program counter. Thus, we improved our algorithm in order to resist any eventual attack that could change the program counter.

5.3.2 A More Secure Unrolled AES Countermeasure

An AES algorithm that resists any round modification attack, must execute the complete operations of 10 rounds (after the initial round) and must verify the number of executed rounds before returning the ciphertext. Otherwise the algorithm must protect the faulty ciphertext from the opponent.

In order to satisfy this requirement, we added two one-byte counters to our AES. The first counter is initialized at 1 then it is shifted one bit to the left at each executing round until the end of the 6th round. The second counter is initialized at 128 or 0x80, then it is shifted one bit to right from the 6th round. Therefore in this algorithm, we have two separate counters that just monitor the encryption before returning the ciphertext, but do not conduct the operations.

In order to have more control over any eventual change in the program, both of the counters are checked before each **AddRoundKey** operation by **RoundTest1** and **RoundTest2** respectively. In addition, the counters are increased separately before and after each **AddRoundKey** operation. Therefore, if any jump happens in the program execution, it will be detected immediately by counter checking. Then, the algorithm sends a fully zero ciphertext. Figure 5.6 shows an overview of this countermeasure.

Besides, in the case of a complex attack on the chip's program counter and both round counters, it will be almost impossible to change all three counters, *i.e.* *RC*, *Counter₁* and *Counter₂* to any other corresponding values in a short action time between two **RoundTest** operations.

```

START
C ← M
Counter1 ← 1
Counter2 ← 128
//ROUND 0
IF ((Counter1=1) & (Counter2=128)) THEN

```

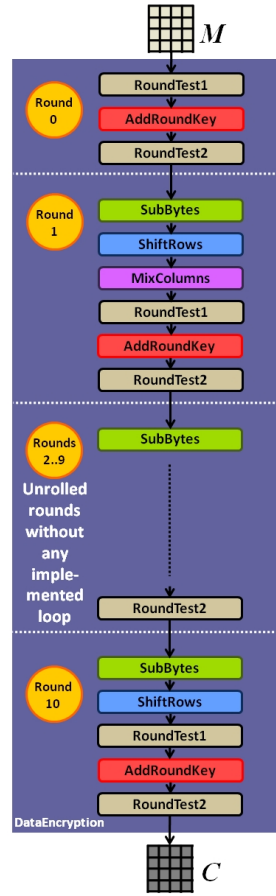


Figure 5.6: Implementation of the unrolled AES with RoundTest1 and RoundTest2 operations.

```

    C ← C ⊕ K[0]
    Counter1 ≪ 1
ELSE
    C ← 0
    RETURN C
ENDIF
//ROUND 1
C ← SBOX(C)
C ← SR(C)
C ← MC(C)
IF ((Counter1=2) & (Counter2=128)) THEN
    C ← C ⊕ K[1]
    Counter1 ≪ 1
ELSE
    C ← 0
    RETURN C
ENDIF

```

```

      :
//ROUND 10
  C ← SBOX(C)
  C ← SR(C)
  IF ((Counter1=128) & (Counter2=16)) THEN
    C ← C ⊕ K[10]
    Counter2 >> 1
  ENDIF
//END OF ENCRYPTION
  IF ((Counter1=128) & (Counter2=8)) THEN
    RETURN C
  ELSE
    C ← 0
    RETURN C
  ENDIF
END

```

Via internal monitoring counters, this countermeasure detects any round modification. If any round modification or any jump occurs, the countermeasure returns a fully zero ciphertext.

Therefore, by using these countermeasures, we can protect our circuit against any round modification attacks by fault injection.

5.4 A Combined Countermeasure against DFA and RMA

In the previous sections of this chapter, we presented our countermeasures against DFA and RMA attacks by fault injection. We can combine these two sets of countermeasures in order to obtain full protection against DFA and RMA attacks by fault injection.

Figure 5.7 shows an overview of the combined countermeasure against DFA and RMA attacks in our AES implementation. We consider **KeyTest** as an inseparable safeguarding operation after each **AddRoundKey**. Therefore, **RoundTest1** is placed before **AddRoundKey** and **RoundTest2** just after **KeyTest**.

5.5 Conclusions

In chapters 3 and 4, we presented several DFA and RMA attacks by laser fault injection. In the current chapter, we presented simple ideas for software countermeasures that protect AES implementation against laser fault attacks. As the fault attack threats are more feasible than normally perceived, the use and the implementation of such countermeasures are mandatory in order to protect cryptographic circuits.

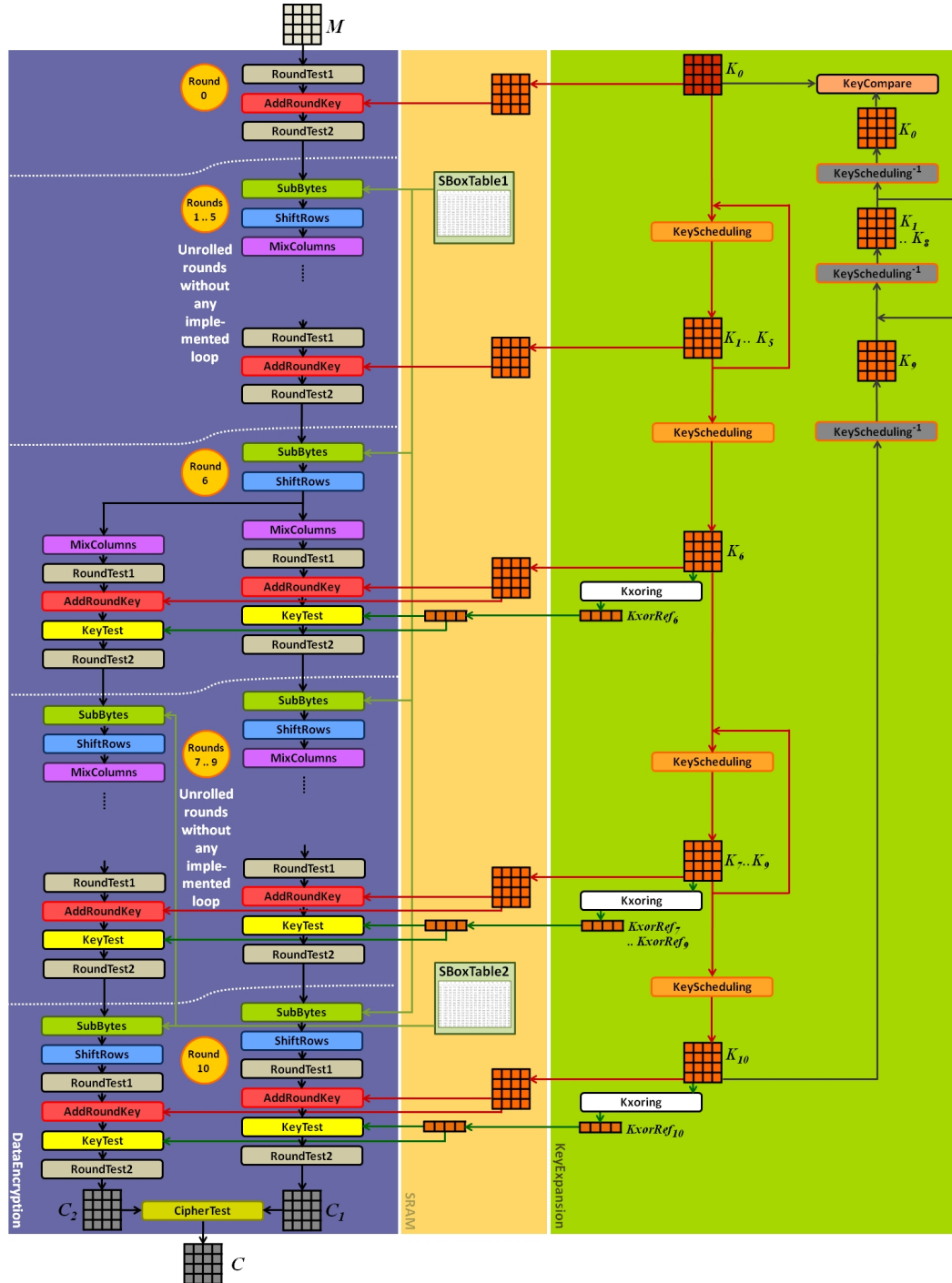


Figure 5.7: Combined countermeasures against DFA and RMA attacks in our AES implementation.

We expect that the presented embedded countermeasures will strongly protect our AES and can be adapted easily to other implementations in order to protect them.

Nevertheless, presented countermeasures consume more circuit resources and increase the encryption time. However, using these countermeasures is a fair trade-off in order to protect sensitive information in an AES cryptosystem by losing some efficiency in execution time and in memory space.

Other Security Perspectives

Contents

6.1	Introduction	133
6.2	A Very Close to Perfect Countermeasure against Power Analysis Attacks	133
6.2.1	Introduction	133
6.2.2	Criteria for a Perfect Solution	134
6.2.3	A Very Close to Perfect Solution	135
6.2.4	Possible Attack on Our Countermeasure and Our New Solution	142
6.2.5	Conclusion	142

6.1 Introduction

In this chapter, we present our other works regarding the security of secure systems.

6.2 A Very Close to Perfect Countermeasure against Power Analysis Attacks

In this section, we describe a solution against power analysis attacks on smart cards. The principle involves making smart card consumption independent from its circuit activity; i.e. cutting the leakage channel.

We show that a smart card's chip can be indirectly powered by using an embedded set consisting of an OLED panel and a photovoltaic cell. In this case, the power consumption of the smart card, observed on its external contact, will be constant and equal to the OLED panel consumption. This countermeasure can be integrated into the plastic shape of future generations of smart cards.

6.2.1 Introduction

In section 1.1, we described *Side Channel Analysis* (SCA) as a principle category of hardware attacks. SCA monitors any hardware information leakage during the encryption operations. This information leakage may be related to sensitive data processing and may reveal the secret key. This idea was first introduced by Paul

Kocher in [Kocher 1996]. Within these attacks, Single and Differential Power Analysis, known as SPA and DPA, are particular analytic techniques exploiting any power consumption variation in order to discover the operations executed by the hardware [Kocher 1998] [Kocher 1999]. Many countermeasures have been developed against these attacks, however their efficiency is still not optimal.

6.2.2 Criteria for a Perfect Solution

Whenever the power supply line of a smart card chip is connected to its external contact, any countermeasure cannot entirely hide the power consumption variations from the attacker. The only solution is to separate the power supply line from its contact. A way of meeting this requirement is to integrate a separate power supply into the card structure. The embedded power supply will need an external power source, provided by the card contact. In addition, it should entirely mask the chip's consumption. It is difficult to find such an electrical source that could fit into the smart card structure.

A set, including a light source and a photovoltaic cell, could be a possibility. The requirement is that the set can satisfy the size, the power and the fast switching requirements of smart cards. Between the light sources, a *light-emitting diode* (LED) or an *organic* LED (OLED) could be good choices to meet these criteria.

6.2.2.1 LED

A light-emitting diode (LED) is a semiconductor diode that emits light when it is forward-biased by its electroluminescence effect. In fact, when an LED is switched on, electrons are able to recombine with holes within the device and release energy in the form of photons.

The wavelength of an LED's light can be in a visible or in an invisible domain (e.g. ultraviolet and infrared). It is determined by the energy gap of the semiconductor material and corresponds to the photons energy level.

LEDs have many advantages over traditional light sources, for instance: better luminous efficiency, longer lifetime, improved robustness, smaller size, faster switching, and greater durability and reliability. Nowadays, they are considered as a good source for general lighting. They are the subject of much research for this purpose.

6.2.2.2 OLED Panel

An OLED, is a light-emitting diode (LED) with an organic electroluminescent layer. In other words, the OLED's electroluminescent layer is composed of a film of organic compounds. When an electric current passes through this layer of organic semiconductor material, it emits light. This layer is formed between two electrodes, where at least one of the electrodes is transparent.

Nowadays, OLEDs are used as display for mobile phones and personal digital assistants (PDAs). They can be thinner and lighter than LCD panels and can achieve

higher contrast ratios, but have a shorter lifetime. There is some research on the use of OLEDs as light sources for general lighting. The first products are being commercialized.

6.2.2.3 Photovoltaic Cells

A photovoltaic cell is a device composed of semiconductor layers. It converts the energy of light directly into electricity by its photovoltaic effect. When the light hits a photovoltaic cell, its semiconducting material absorbs several photons. They generate electron-hole pairs (and sometimes heat). The electrons flow through the semiconducting material. Due to the special composition or the architecture of photovoltaic cells, the electrons are only allowed to move in a single direction. So, they create an electrical field between the two edges of the photovoltaic cell. This difference corresponds to the direction of the current. The produced electricity can be obtained as a direct current from the two edges.

Contemporary technology in the domain of commercialized photovoltaic cells has an efficiency of about 20%. Besides, different research has lead to the discovery of new materials and new cell architectures with an efficiency that sometimes reaches more than 40%. The most efficient photovoltaic cells are often used for satellite or military purposes and are not available on the retail semiconductor market.

6.2.3 A Very Close to Perfect Solution

Our solution consists of using an OLED panel as a light emitter and a photovoltaic cell as a power supply for the smart card's microcontroller. So, without any change in the card reader, it will still provide the electrical power to the card's contact. However, in the new card structure, the V_{cc} line must only be connected to the OLED panel. Then, the electrical energy for the smart card's microcontroller will be supplied by the photovoltaic cell .

To implement our solution, we searched among available technologies on the market. For the light emitter, we were looking for something as small as a smart card with good efficiency. We found many single and array LEDs, some OLED displays for mobile and PDAs, but only one available commercially as an OLED panel for illumination: The Osram Orbeos CDW-031 OLED [Semiconductors 2009]. It is a thin panel as shown on the left side of figure 6.1. The OLED layer, without its protective glass, has a thickness of only several hundred nanometers. Some of its technical information is shown in table 6.1 [Semiconductors 2009]. We also chose some LED arrays for our experiments.

For the photovoltaic cell, we found many models. However, given our restrictions, we had very few choices for LED arrays and OLED panel surfaces with good efficiency. We selected some models as described in table 6.2 because of their higher

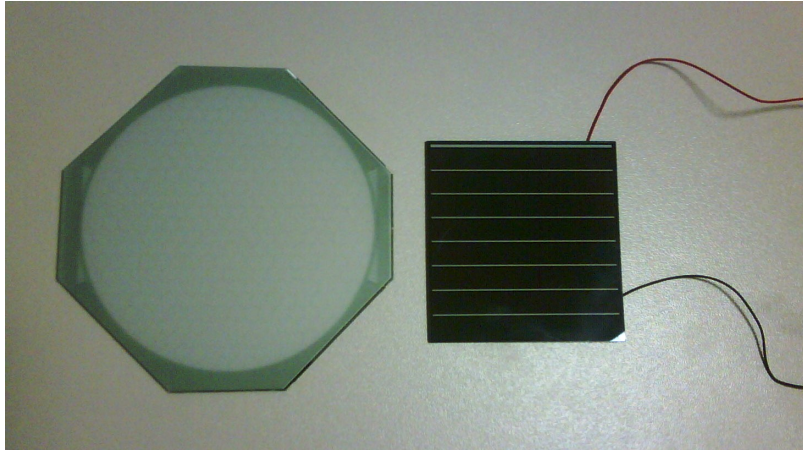


Figure 6.1: An Osram Orbeos OLED panel and a Sanyo Amorton AM-8801 photo-voltaic cell.

Table 6.1: Some technical information about the Osram Orbeos CDW-031 OLED panel.

Parameters	Value	Unit
Diameter of light output area	79	<i>mm</i>
Forward voltage (maximum value)	4.5	<i>V</i>
Power consumption (maximum value)	0.71	<i>W</i>
Luminance (for a forward current of 186 <i>mA</i>)	1000	<i>Cd/m²</i>
Luminous efficacy (typical value)	23	<i>lm/W</i>

power values. For two models with small dimensions, we obtained a quantity of cells that cover the LED arrays or the OLED's light output surface.

In order to find the most efficient photovoltaic cell, at the first, we tested output of different cells with LED arrays and OLED panel. Since results showed improved efficiency with the OLED panel, we focused on using it for the following tests.

Then, we performed a test to measure the output of different cells for 5 different levels of power on the OLED panel. We turned it on to its maximum authorized power and marked this level as our first measurement. Then, we boosted the input level by between 2 and 8.5 times for the following level measurements. To obtain more precise results, we applied 9 different resistance levels from $470\ \Omega$ to $1\ M\Omega$ to the cells' output at any level of light. Table 6.2 shows the maximum output power and the applied resistance for each photovoltaic cell. Finally, we chose Sanyo Amorton AM-8801 [Sanyo 2008] (shown on the right side of figure 6.1) because it was more efficient than any other.

Boosting the OLED panel during the test always increased the output, but the efficiency did not improve after the third power level. In fact, it even decreased quickly. Figure 6.2 shows the output and the efficiency curves when we applied a 1000Ω resistance to the output of the AM-8801 photovoltaic cell.

Table 6.2: Maximum output power of photovoltaic cells when exposed to Osram Orbeos OLED panel light, boosted to $8.25W$.

Manufacturer	Model	Effective area for each cell ($mm \times mm$)	Number of serially connected cells	Output power (mW)	Applied resistance (Ω)
Sanyo	Amorton AM-1437	27.8×8.4	14	11.80	100000
Sanyo	Amorton AM-8801	54.3×53.0	1	21.16	1000
Solarex	MSX-005F	95.8×57.0	1	5.17	1500
Solems	07/048/016	48.0×16.0	6	9.19	100000
Taizhou Lead Strong	LS60 $\times$ 60-4M150	50.0×50.0	1	8.17	470

Therefore, we implemented our countermeasure by connecting the V_{cc} input of the card's microcontroller directly to the photovoltaic cell's output. The current supplied by the photovoltaic cell could successfully turn on generic smart cards whose supply voltage can vary from 3 to 5V. Nonetheless, there was not sufficient stability to turn on an EMV (Eurocard, Mastercard, Visa) chip card.

According to [ISO/IEC 7816-3 2003], the EMV cards may have a protection against voltage variations that disables them when the V_{cc} is more than 10% below or above its typical value. So, we used an Infineon TLE 4264 as a 5V low drop fixed voltage regulator to maintain voltage stability. This voltage regulator needs two capacitors at entry and exit in order to establish a stable voltage of 5V. We used a $100\mu F$ electrolytic capacitor at input and a $10\mu F$ electrolytic capacitor at output for testing different cards. Figure 6.3 shows a schematic of our countermeasure.

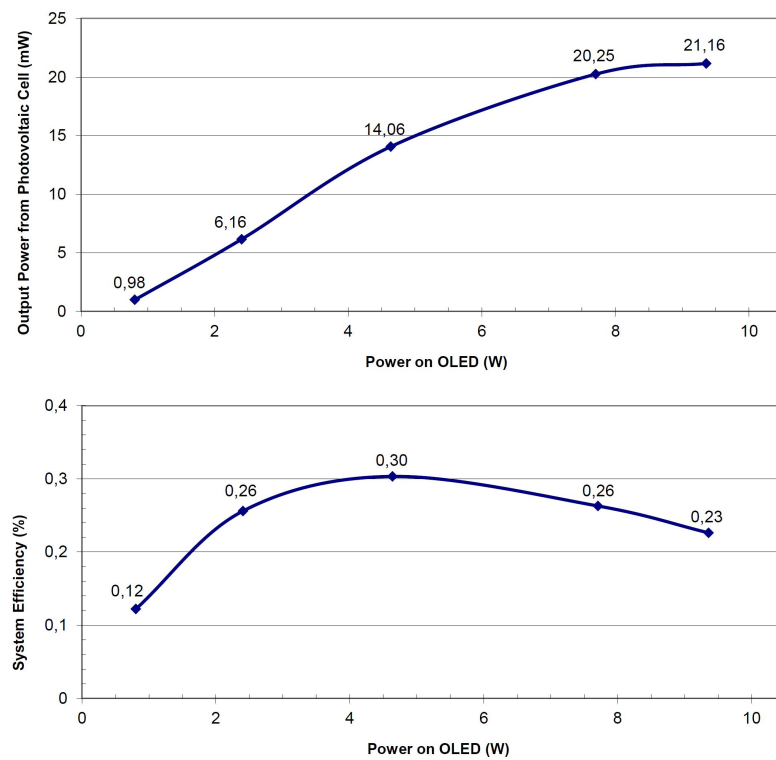


Figure 6.2: Output and efficiency of the system by boosting the OLED panel over its authorized power limits.

We can consider the set containing the OLED panel, the photovoltaic cell, the voltage regulator, the two capacitors and the card's chip as a *protected system* that must hide power consumption variations from the external world. We built a card adapter by soldering the protected system elements and a smart card connector onto a Krystal Universal Card, as shown in figure 6.6. Consequently, when we insert a smart card into the reader, the set operates the prototype of a future generation secure card which is protected against power analysis attacks.

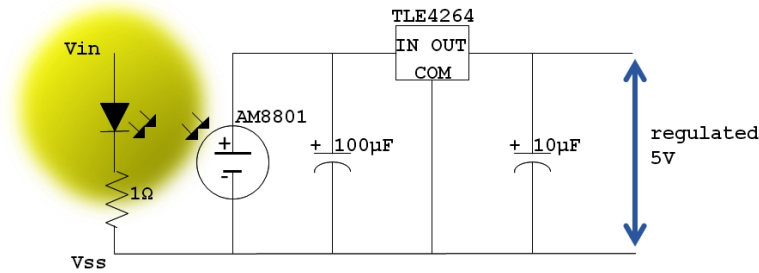


Figure 6.3: Electronic schematic of the countermeasure.

We applied our countermeasure to different protected cards. We could successfully turn on and receive a correct ATR (Answer To Reset) on four different cards using a photovoltaic cell, a voltage regulator and two capacitors. The best results were achieved using two recent EMV chip cards, one supplied by GyD Iberica under the reference 05/09 16953410 and the other one supplied by Sagem Orga under the reference 04/10 103043-1.

We boosted the OLED panel input power to approximately 3.5 times higher than its maximum authorized power in order to turn on these cards. This power is also about 7.6 times higher than the maximum standard value for class A cards (i.e. 5V cards) [ISO/IEC 7816-3 2003]. ISO/IEC 7816-3 permits a current up to 60mA maximum and a voltage that could be up to 10% higher than the typical 5V. However, as the older cards have higher power consumption, they could not be turned on by the limited energy of our photovoltaic cell. For these cards, we boosted the OLED panel to approximately 16 times higher than its maximum authorized power. Table 6.3 shows these results.

Table 6.3: Results for different smart cards using the energy of the photovoltaic cell.

Card Type	Manufacturer	Reference No.	V_{in} on OLED (V)	Power on OLED (W)	V_{CC} supplied by protected system (V)	ATR
EMV chip card	Gemplus	04/03 46156	4.35	4.13	5.042	✓
EMV chip card	GyD Iberica	05/09 16953410	4.08	2.51	5.042	✓
EMV chip card	Oberthur	05/07 45785	5.95	12.19	1.930	
EMV chip card	Oberthur	01/08 47576	5.75	11.78	5.042	
EMV chip card	Oberthur	06/08 49064	5.95	12.19	2.113	
EMV chip card	Sagem Orga	04/10 103043-1	4.09	2.74	5.042	✓
French health card	Sagem DS	07/2007	4.78	5.83	5.042	✓

To perform our tests, we used a Smartware UltraSmart X-CORE T series card reader. The reset commands were sent by the SMarT-I (described in subsection 2.1.1) to the card reader.

Figure 6.8 shows that after applying our countermeasure, the power consumption signal on the V_{cc} contact of the reader is entirely flat and does not show any variation. So, any power analysis attack on the reader becomes impossible.

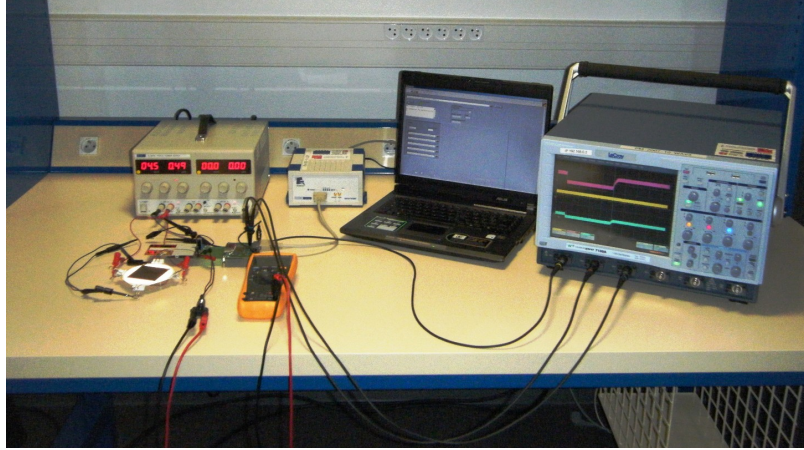


Figure 6.4: The test bench.

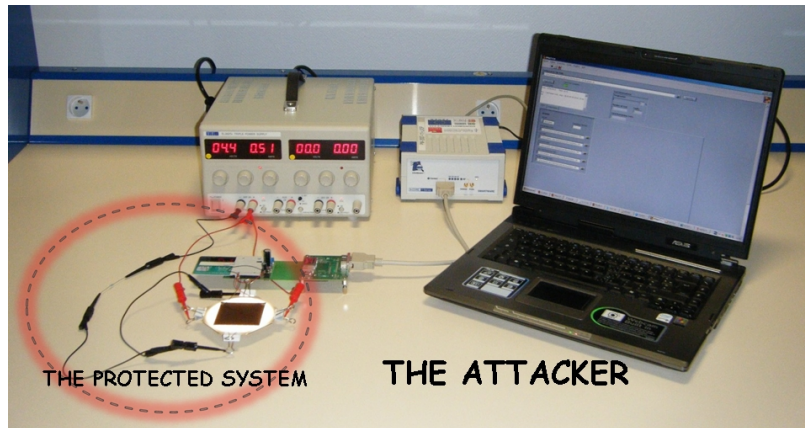


Figure 6.5: The protected system and the attacker.

The minimum value of the input capacitor is a function of photovoltaic output power and card consumption. For the GyD Iberica card, the value of the input capacitor can be reduced to $50\mu\text{F}$ in order to successfully respond to a single reset command. However, when a second reset command is immediately sent, the voltage regulator could not maintain the regulated voltage at $5V$ and the card did not respond. To continuously send commands, the value for this capacitor in our tests needed to be at least $70\mu\text{F}$.

More experiments showed that the necessary input power on OLED can be reduced by using higher value for the input capacitor. For instance, by using two capacitors of $100\mu\text{F}$ instead of only one with the Gemplus 04/03 46156 model, the card only needs an input power of 3.29mW on the OLED rather than the initial value of 4.13mW .

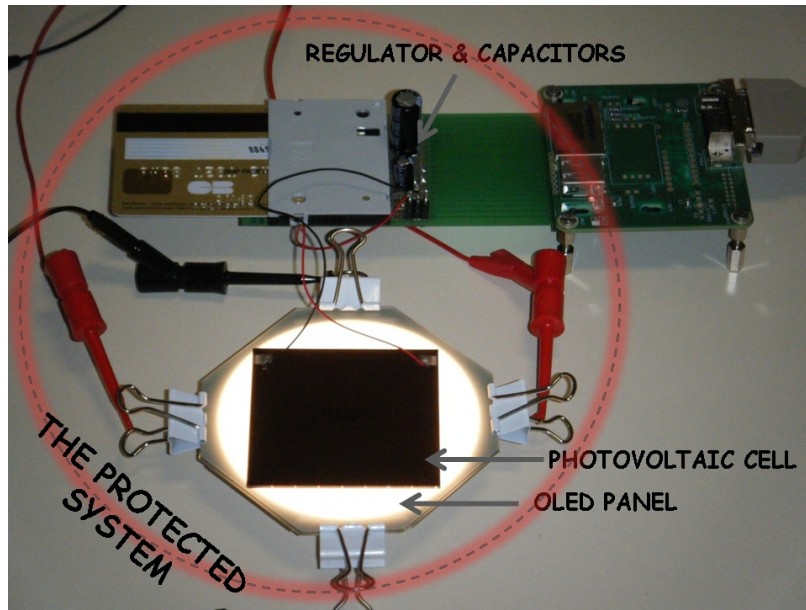


Figure 6.6: Close-up of the protected system.

Thus, we could build our countermeasure against power analysis attacks. As the set of the OLED panel and the photovoltaic cell has a thickness of a few hundred microns without their protective glass, they can be integrated into the plastic shape of future generations of smart cards. In this case, the energy wasted in the glass will also be eliminated. Figure 6.7 shows a close-up of the thickness of the OLED panel and the photovoltaic cell with their protective glass and a smart card.

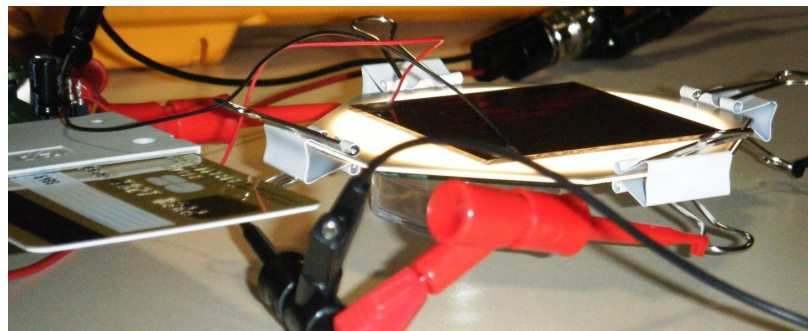


Figure 6.7: Close-up of the thickness of the OLED panel and the photovoltaic cell (both with their protective glass).

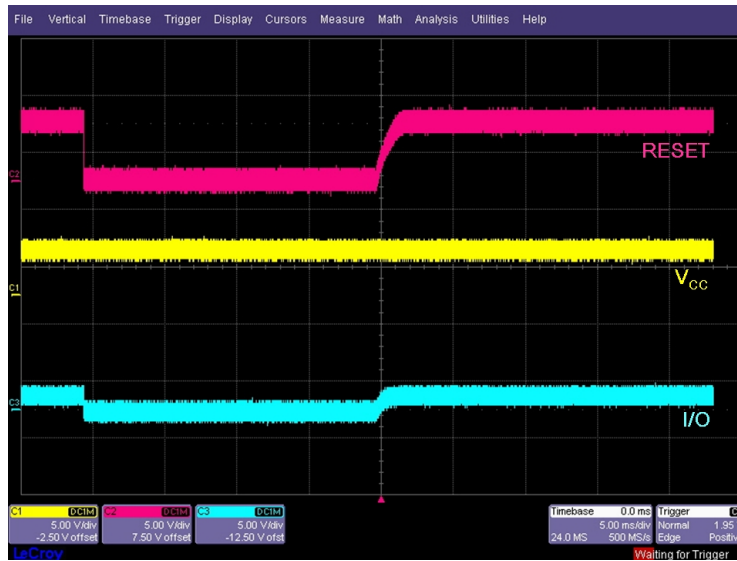


Figure 6.8: Reset, power consumption and I/O signals observed on the card contacts at the reset moment.

6.2.4 Possible Attack on Our Countermeasure and Our New Solution

Normally, we should connect all the ground connections together. This configuration can induce a new potential power analysis attack on the card's ground contact. The attacker might use an embedded resistance in the card reader in order to see the chip's consumption signals. If he puts this resistance between the ground contact of the card and the ground line of the card reader, he might visualize the chip's consumption variations.

To overcome this problem, we found a solution which involves separating the two ground lines. As shown in figure 6.9, we considered a twisted pair connection between the I/O and the clock contacts of the card and the reader. The principle resembles the implementation of two data lines, namely D+ and D-, in an Universal Serial Bus (USB) that establishes a direct data transfer connection between two components without using a common ground line.

6.2.5 Conclusion

We demonstrated our solution for protecting a smart card against power analysis attacks. A smart card can be powered on indirectly by using an OLED panel and a photovoltaic cell. Thus, the card's power consumption on its external contact will be permanently constant and equal to the OLED panel consumption. Besides, it is necessary to separate the chip's ground line from the external ground contact. This countermeasure can be integrated in the card's plastic shape in future generations of smart cards.

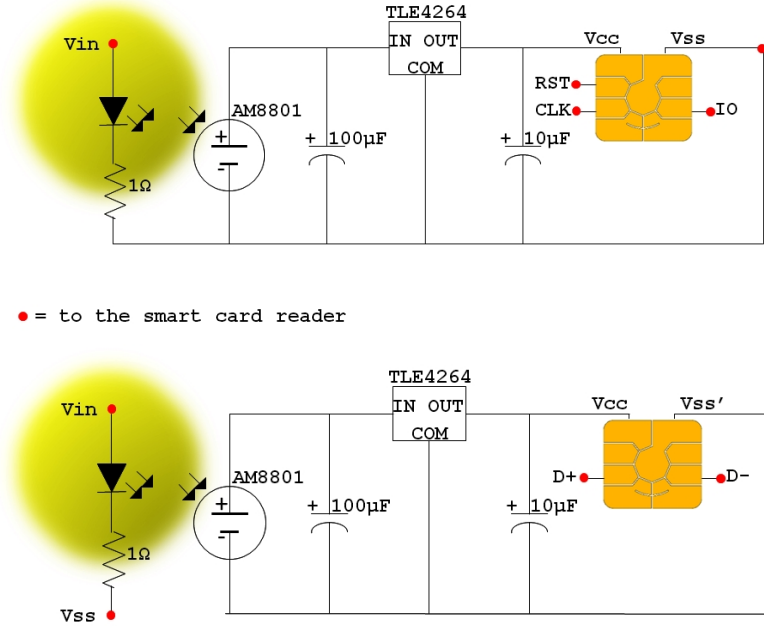


Figure 6.9: Electronic schematic of the countermeasure and the card contacts, before and after applying the new solution.

Our countermeasure has a power consumption about 7.6 times greater than ISO/IEC 7816-3 standard limits. It also has an excessive heat dissipation. Since more than 41.2% of our OLED panel was not covered by the photovoltaic cell, the consumption could be reduced to about 58.7% of the current value as long as the size of the OLED panel is adapted correctly to the size of photovoltaic cell or vice versa.

In addition, by boosting our OLED panel above its authorized power, the system output increases but the system efficiency decreases after a short period. So, by finding a more efficient panel, the problem of over consumption and heat dissipation may be resolved.

It seems that our Osram Orbeos panel is the only commercialized OLED at the present time. The OLED panels which should be coming onto the market in the next few months, will have better efficiency. For example: the first commercialized Konica Minolta OLED panel that arrives in a short time will have a light emission efficiency of about 64 lm/W ; i.e. 278% more than our current panel.

For commercial use of our countermeasure, some other technical criteria should be taken in account. It is especially important that the ISO/IEC 7816-3 specifications for the authorized values of other electrical parameters in a smart card and also the card's response time restrictions be considered in the design.

However, our countermeasure can not prevent electromagnetic attacks (EMA). Theoretically, wires might act as antennae and conduct a negligible EMA residue.

Nonetheless, in summary, our solution is a countermeasure very close to perfect against power attacks because it can completely hide power consumption variations.

Conclusions and Perspectives

The objective of this thesis was the identification and the study of laser fault injection threats on cryptographic systems.

In the first part, we examined the practical feasibility of some theoretical models of fault attacks. We performed laser fault injections on a microcontroller implementing an AES cryptographic algorithm. Differential fault Analysis (DFA) is a fault injection method used to reveal a secret key by comparing the correct and the corresponding faulty results. Theoretical models of DFA on AES mostly require the injection of single-byte or single bit faults. In our experiments, the laser beam created some additional faults when compared to the required models. We succeeded in excluding the logical effect of additional faults by temporal and spatial accuracy in fault injection. Moreover, by using a classification method for the faults, we proposed an extended version of Giraud's single-bit DFA.

Then, we extended our research to explore new models of fault attacks. Symmetric cryptographic algorithms often consist of repeated rounds. Fault attacks on the round counter or on the reference number of rounds have been much less studied than DFA attacks. With the precision obtained in the first part of our studies, we developed a couple of new attacks with their associated cryptanalytic solutions. They have two targets: the round counter and the reference number of AES rounds. More than ten of these attacks are based on simple scenarios, including an attack on one of the two aforementioned targets (RC or R_{max}), at a specific time of algorithm execution.

Our experiments warn us of the feasibility of the described attacks in the literature by laser. Our tests demonstrate that single-byte or single-bit attacks are still feasible with a laser beam that hits additional bytes on the circuit when the laser emission is accurate and associated with other techniques. Therefore, the implementation of appropriate countermeasures is necessary in the design of new circuits. For these reasons and in order to finalize our work, we proposed and evaluated some countermeasures against these attacks.

Moreover, being interested in other subjects of cryptosystem security, we participated in a few other related research. We developed a countermeasure against power analysis attacks on smart cards. This countermeasure is composed of an organic light emitting diode (OLED) and a photovoltaic cell in order to make the card's consumption independent from the processed data. The OLED panel is a few hundred nanometers thick and the photovoltaic cell has a thickness of several

tens of micrometers. Therefore, this set can be implemented in the interior of a future generation of smart cards and protect them due to the constancy of OLED cell consumption. As part of another project, we participated in the evaluation of a countermeasure against fault attacks. We examined its resistance by temperature violations.

Prospects for continuing this research are:

- **Research on other fault injection mechanisms:** The study of fault injection on microcontrollers; including the arithmetical logic unit (ALU), the bus and the key scheduling operations.
- **Study of flash memory vulnerability:** Including research on changing the loaded program using faults and fault injection on the initial values stored in flash memories.
- **Implementation and validation of countermeasures:** How to protect the circuits from fault injection? and from the release of sensitive information using faults?

We hope to continue these perspectives in future research.

Résumé étendu en français (French Extended Abstract)

Contents

8.1	Introduction	148
8.1.1	Attaques en fautes des systèmes cryptographiques	149
8.1.2	Attaques en fautes sur AES	155
8.1.3	Physique de l'injection de fautes par laser	156
8.2	Caractérisation sécuritaire	157
8.2.1	Plateforme de caractérisation sécuritaire MircoPackS	157
8.2.2	Caractéristiques du banc laser	158
8.2.3	Préparation de circuit et des échantillons	159
8.2.4	Caractérisation sécuritaire : Première cartographie de la susceptibilité d'injection de fautes	162
8.2.5	Exploration de notre circuit	163
8.3	Pratique de la DFA par laser sur AES	164
8.3.1	Problématiques de la pratique de la DFA par laser	164
8.3.2	DFA mono-octet de Piret et Quisquater	164
8.3.3	DFA mono-bit de Giraud	164
8.3.4	Résultats pratiques des DFA mono-octets et mono-bit	166
8.3.5	Une DFA multi-octets étendue	166
8.4	Attaques par modification de rondes	167
8.4.1	Modèles faisables d'attaque sur notre AES	168
8.4.2	Modèle d'attaques faisables sur notre AES	170
8.5	Contre-mesures	174
8.5.1	Contre-mesures contre l'analyse différentielle de fautes	174
8.5.2	Contre-mesures contre l'analyse de modification de rondes	174
8.5.3	Une contre-mesure combinée contre DFA et RMA	176
8.6	Autres perspectives de sécurité	178
8.6.1	Une contre-mesure efficace contre les attaques d'analyse de consommation	178
8.7	Conclusions et perspectives	180

8.1 Introduction

La cryptographie est l'étude et la pratique des méthodes pour écrire des messages secrets. Son objectif est de cacher le contenu des messages à tout le monde sauf à un ou quelques destinataires désignés qui seront les seuls à pouvoir enlever le secret et à comprendre le sens du message [Mollin 2007] [Paar 2010].

La cryptographie moderne est basée sur les mathématiques, l'informatique et le génie électrique. Elle comprend des méthodes symétriques et asymétriques. Dans la première famille, les messages sont chiffrés et déchiffrés en utilisant une clef unique (la clef secrète). Par contre, dans les méthodes asymétriques, le chiffrement et le déchiffrement s'effectuent en utilisant deux clefs séparées : une pour le chiffrement des messages par l'expéditeur et l'autre pour le déchiffrement par le destinataire.

Selon [Denning 1983] et [Van Tilborg 2005], un *système cryptographique* est un système de chiffrement et de déchiffrement des messages, composé d'un *algorithme de chiffrement*, un *algorithme de déchiffrement* et un triplet bien-défini sur des espaces :

1. $\mathcal{M}$: espace des textes clairs.
2. $\mathcal{C}$: espace des textes chiffrés.
3. $\mathcal{K}$: espace des clefs.

L'algorithme de chiffrement $\mathcal{E}$ transforme tous les textes clairs en textes chiffrés en utilisant la clef donnée :

$$\mathcal{E}_{K_E} : \mathcal{M} \rightarrow \mathcal{C} \quad \text{où} \quad K_E \in \mathcal{K}$$

L'algorithme de déchiffrement $\mathcal{D}$ aussi assigne tous les textes chiffrés en utilisant la clef correspondante aux textes clairs initiaux :

$$\mathcal{D}_{K_D} : \mathcal{C} \rightarrow \mathcal{M} \quad \text{où} \quad K_D \in \mathcal{K}$$

En cryptographie asymétrique, la clef utilisée pour le déchiffrement est différent de la celle du chiffrement. Par ailleurs, la cryptographie symétrique emploie une clef unique pour les deux transformations : $K_E = K_D$.

Les transformations de chiffrement et de déchiffrement sont généralement déterminées par des fonctions *injectives*.

Les systèmes cryptographiques sont également appelés *systèmes de chiffrement* ou *crypto-systèmes*. Dans les modèles primaires, les opérations ont été effectuées par des méthodes manuelles en utilisant le crayon et le papier. Au 18^{ème} siècle, les méthodes mécaniques ont été introduites pour accélérer la cryptographie. Environ un siècle plus tard, elles ont été remplacées par une nouvelle génération de machines électromécaniques. La machine allemande Enigma et l'américaine ECM Mark II étaient deux modèles célèbres de ces crypto-machines utilisées au cours de la Seconde

guerre mondiale. Aujourd'hui, le chiffrement et le déchiffrement ont évolué et sont sortis de l'ère mécanique : ils sont souvent réalisés par des solutions logicielles ou accélérateurs matériels.[Konheim 2007]

Toutefois, dans l'ère moderne, la sécurité d'un crypto-système est toujours basée sur la sécurité de la clef du déchiffrement. Cela a été d'abord désigné dans une des principes d'Auguste Kerckhoffs, présentées en 1883 : “ Un crypto-système devrait être sûr si tout le système, sauf la clef, est à la connaissance du public” [Kahn 1973].

Par conséquent, aujourd'hui les crypto-systèmes contiennent des clefs secrètes utilisées par leurs algorithmes de cryptographie. Pour cette raison, ils sont le sujet de plusieurs recherches afin de faire progresser leur sécurité et leur résistance contre toute manipulation non-autorisée.

8.1.1 Attaques en fautes des systèmes cryptographiques

Deux principales familles d'attaques contre les systèmes cryptographiques existent :

1. Attaques mathématiques et cryptanalytiques

Elles recherchent des faiblesses dans les schémas ou les algorithmes de chiffrement afin d'en déduire les clefs secrètes à l'aide de méthodes mathématiques. Si un attaquant n'arrive à trouver aucune faiblesse dans un crypto-système pour l'exploiter par une attaque cryptanalytique, il sera toujours possible d'effectuer une *recherche exhaustive* de la clef [Paar 2010]. Toutefois, cette recherche ne donnera jamais une réponse dans un temps de calcul acceptable pour une taille de clef bien choisie par rapport aux capacités de calcul actuelles.

2. Attaques matérielles

Cette grande famille d'attaques s'applique à des composants matériels (circuits intégrés, mémoires, etc.) et comprend principalement les deux sous-catégories suivantes :

- (a) Analyse des canaux cachés

Ces attaques sont basées sur l'analyse de toutes les fuites d'information liées au fonctionnement d'un circuit ; soit par l'analyse de sa consommation [Kocher 1996], soit par l'analyse de son rayonnement électromagnétique [Gandolfi 2001] [Joye 2005], soit par l'analyse de son temps de réponse pour extraire des données qu'il manipule [Van Tilborg 2005], etc.

- (b) Attaques en fautes

Elles consistent en la modification volontaire de l'environnement de la puce et l'altération de son fonctionnement par différentes manières ; puis en l'exploitation des comportements ou des résultats fautés afin d'en déduire des informations secrètes. En 1997, [Bellcore 1996] et [Boneh 1997] ont montré la possibilité d'extraire des informations secrètes par des perturbations physiques. Ensuite, [Biham 1997] a présenté une méthode

d'analyse différentielle pour exploiter des fautes. Ces attaques peuvent être mises en place en utilisant différentes techniques physiques communiquées en [Tria 2000]. Aujourd'hui, différentes méthodes d'analyse ont été développées pour révéler des informations secrètes à partir des comportements ou des résultats fautés.

Ce travail de thèse est consacré à l'étude des attaques en fautes : Comment les fautes pourront être injectées et comment les résultats fautés pourront être exploités et mettre en danger le secret des systèmes cryptographiques ?

8.1.1.1 Définition de faute

Une faute dans un système cryptographique fait référence à un événement accidentel ou intentionnel qui provoque l'échec de l'exécution correcte du processus de chiffrement ou de déchiffrement. Dans ce cas, le système cryptographique peut fonctionner anormalement et un résultat de chiffrement ou de déchiffrement incorrect, considéré comme *fauté* peut être obtenu.

Une exécution ou un résultat fauté est considéré comme *reproductible* s'ils peuvent être reproduits régulièrement dans les mêmes circonstances.

8.1.1.2 Systèmes sécurisés et non-sécurisés

Le bon fonctionnement des circuits intégrés classiques dépend de leurs conditions physiques. En plus, ils n'empêchent pas la fuite des informations qui permettent à découvrir leurs contenus par l'analyse des canaux auxiliaires. Ces types de circuits intégrés sont considérés comme *non sécurisés* et ne conviennent pas pour des calculs cryptographiques.

Par ailleurs, il existe des circuits intégrés spécifiques pour les systèmes cryptographiques qui résistent à des conditions extrêmes physiques. Dans ce type de circuit, quand il y a un risque de mauvais fonctionnement, comportement ou résultat, leurs opérations s'interrompent. Ils peuvent être parfois pré-programmés pour donner un résultat intentionnellement faux qui ne permet pas de déduire des informations sensibles. Ces circuits sont appelés des *systèmes sécurisés*. Ce genre de protections peut s'appliquer aux fuites des canaux auxiliaires.

Par exemple, une carte bancaire à puce intégrée est un système sécurisé. Selon les normes ISO/IEC 7816-3, les opérations de la puce s'interrompent si la tension V_{CC} du circuit atteint une augmentation ou diminution de plus de 10% de sa valeur originale de 5,0 V [ISO/IEC 7816-3 2003].

8.1.1.3 Différents types de faute

Les fautes affectant les circuits intégrés peuvent être classées selon trois catégories en fonction de leur persistance :

- **Fautes provisoires ou transitoires** : Cette catégorie comprend les fautes temporaires ou à court terme. Elles disparaissent après l'arrêt de l'injection de fautes. Donc, après l'écoulement d'un certain temps, la puce retrouve un fonctionnement normal sans besoin de réinitialisation. Par exemple, le chauffage d'un circuit crée des fautes par la prolongation des temps de propagation ; mais, après la diminution de la température, le circuit reprend son fonctionnement correct.
- **Fautes permanentes** : Les fautes permanentes sont persistantes, mais réversibles. Elles disparaissent après une réinitialisation du circuit. Donc, elles ne sont pas destructives et n'endommagent pas le circuit de façon définitive. Par exemple, une faute injectée sur une cellule de SRAM persiste jusqu'à la réécriture de son contenu ou de la réinitialisation du circuit. Par ailleurs, une faute injectée sur un code de programme stocké dans une mémoire non-volatile a un effet plus persistant. Dans ce cas, la réinitialisation du circuit n'est pas suffisante pour la récupérer du contenu de la mémoire et une reprogrammation est exigée.
- **Fautes destructives** : La pratique de certaines méthodes d'injection de fautes peuvent créer des dommages perpétuels sur le matériel. Une fois infligée, ces destructions peuvent affecter le comportement de la puce de façon permanente. Par exemple, une émission laser avec un niveau d'énergie élevé sur une zone de mémoire peut définitivement détruire certaines cellules du stockage de données. Dans ce cas, les cellules endommagées de la mémoire ne peuvent être plus ni réécrites, ni récupérées avec la réinitialisation, voire avec la reprogrammation du circuit.

8.1.1.4 Méthode d'analyse de fautes

Plusieurs méthodes d'analyse différentielles ont été découvertes par des chercheurs pour les attaques en fautes. Ces méthodes d'analyse nécessitent généralement l'injection de fautes transitoires ou permanentes.

- **Réduction de rondes ou Round Reduction (RR)** : Consiste en la réduction du nombre des étapes répétitives d'un algorithme afin de faciliter la cryptanalyse de son résultat de chiffrement. Cette méthode a été présentée par Hamid Choukri et Michael Tunstall dans [Choukri 2005].

De nombreux algorithmes cryptographiques sont en effet basés sur la répétition de séquences identiques de transformations, appelées *rondes*. Une partie significative de la sécurité de ces algorithmes contre la cryptanalyse est basée sur la répétition de rondes. Donc, toute diminution du nombre de rondes réduit leur sécurité.

La réduction de rondes appartient à la famille des attaques par modification d'algorithme. Par exemple, supposons une attaque qui fait un saut après

l'exécution de quelques instructions au début de l'algorithme ou peu après la première ronde à la fin. Ainsi, les rondes restantes sont ignorées et le chiffré final sera le produit d'une petite partie des processus d'algorithme et pourrait révéler facilement la clef. Dans ce cas, la cryptanalyse sera très rapide et facile. Sa complexité ne correspond plus à la cryptanalyse de l'exécution correcte de l'ensemble de plusieurs rondes. Cette méthode a été d'abord présentée dans [Choukri 2005].

- **Analyse différentielle de fautes** ou **Differential Fault Analysis (DFA)** : Cette méthode est basée sur l'injection de fautes pendant le processus du chiffrement afin d'obtenir des chiffrés fautés. Une partie ou toute la clef secrète peut ensuite être retrouvée en comparant les chiffrés fautés et corrects correspondants. Elle est illustrée figure 8.1.

La première alerte sur la possibilité d'exploiter des fautes a été signalée par Bellcore (Bell Communications Research, Inc.) dans [Bellcore 1996] et [Boneh 1997]. La première méthode d'analyse différentielle de fautes, ou DFA a été présentée dans [Biham 1997]. Ensuite, de nombreuses autres méthodes de DFA ont été publiées par des chercheurs sur des différents algorithmes cryptographiques.

Nous présenterons deux méthodes de DFA sur AES dans la partie 8.3.

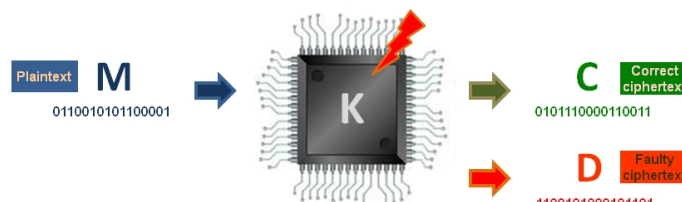


Figure 8.1: Principe de l'analyse différentielle de fautes.

- **Analyse safe-error** ou **Safe-Error Analysis (SEA)** : Cette méthode, d'abord présentée dans [Yen 2000], recherche l'existence de toute différence de comportement d'un circuit en présence de fautes au lieu d'exploiter le chiffré fauté. Une attaque en fautes, peut libérer une alarme interne ou d'arrêter les opérations. Ces signes d'une différence de comportement par rapport à une exécution normale pourront conduire à découvrir des secrets contenus dans un circuit [Blömer 2003].

Les méthodes SEA sont toujours appliquées à des vulnérabilités très spécifiques. En associant les principes de la SEA et le traitement probabiliste de DPA, [Robisson 2007] présente une approche plus réaliste.

- **Analyse différentielle du comportement** ou **Differential Behavioral Analysis (DBA)** : Ce procédé est basé sur la corrélation d'un modèle fonctionnel paramétrée par la valeur d'une clef partielle au comportement du circuit en présence de fautes. Cette méthode d'analyse, présentée dans [Robisson 2007] combine la SEA et l'analyse différentielle de consommation (DPA).
- **Analyse de la sensibilité de fautes** ou **Fault Sensitivity Analysis (FSA)** : Cette nouvelle méthode présentée dans [Li 2010] est basée sur l'analyse de l'état critique où une sortie fautée commence à montrer quelques caractéristiques détectables.

Selon [Li 2010], ces caractéristiques sont liés à la manipulation des données sensibles et pourront être utilisées pour récupérer la clef secrète. Par exemple, l'augmentation de la fréquence d'horloge représente la valeur d'un état critique dès que les fautes viennent d'apparaître. Donc, cette méthode exploite les conditions sensibles d'apparition de fautes au lieu des résultats fautes et corrects correspondants dans la méthode DFA.

8.1.1.5 Techniques d'injection de fautes

Il existe plusieurs techniques d'injection de fautes afin de modifier intentionnellement le fonctionnement correct d'un circuit. Cependant, la réussite de ces méthodes dépend de nombreux paramètres, notamment le niveau de compétence technique et des moyens financiers de l'attaquant ; ainsi que sa compétence sur l'exploitation et l'analyse du comportement et des fautes obtenues [Bar-El 2006] [Barengi 2010]. la figure 8.2 donne un aperçu des techniques les plus connues pour l'injection de fautes :

- La variation transitoire (glitch) de la tension d'alimentation et la modification de l'horloge peuvent être appliquées aux pattes correspondantes sur le circuit.
- L'augmentation de la température et des impulsions électromagnétiques peuvent être appliquées sans décapsulation du circuit.
- L'émission de lumière intense et le rayonnement laser sont des méthodes qui nécessitent la décapsulation du circuit.

8.1.1.6 Différents modèles de fautes

Il existe différents modèles de fautes sur des circuits. Pour bien illustrer les différences, nous considérons $T_1 = \{b_1, b_2, \dots, b_n\}$ comme un ensemble arbitraire de valeurs initiales pour des bits ciblés. On suppose $T_2 = \{b'_1, b'_2, \dots, b'_n\}$ comme l'ensemble des valeurs de bits correspondants à T_1 après une attaque en fautes. Maintenant, nous révisons l'effet de différents modèles de fautes sur les bits ciblés :

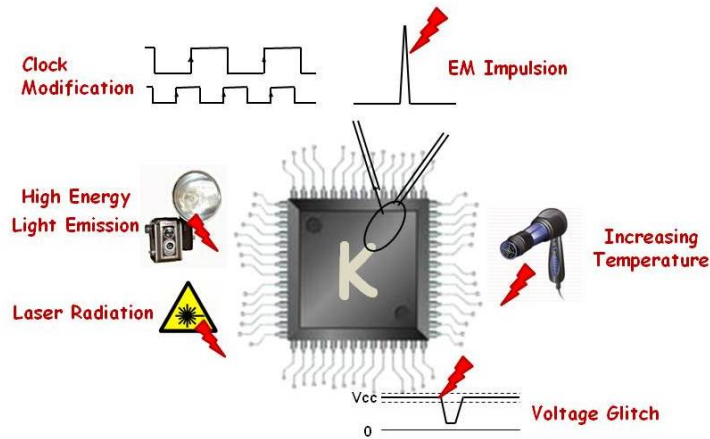


Figure 8.2: Aperçu des principales techniques d'injection de fautes.

- **Basculement ou inversion de bit** : Quand les valeurs de bits ciblés sont changés à leurs valeurs opposées ; soit si et seulement si :

$$\forall i : 0 \leq i \leq n \quad ; \quad b'_i = 1 - b_i \quad (8.1)$$

- **Collage (Stuck-at fault)** : Dans ce modèle de faute, les bits ciblés sont fixé en permanence à leur valeur précédente. Par conséquent, quand de nouvelles valeurs doivent être affectées aux bits ciblés, l'opération d'écriture de mémoire échoue. Cet effet est généralement considéré comme une faute destructive en raison d'un fil, une grille ou une cellule de la mémoire endommagés ; mais, il pourrait être également une faute permanente qui disparaisse après une réinitialisation du circuit.

Le modèle de faute est considéré *collé à 0*, si et seulement si :

$$\forall i : 0 \leq i \leq n \quad ; \quad b_i = b'_i = 0 \quad (8.2)$$

Autrement, le modèle de faute est considéré *collé à 1*, si et seulement si :

$$\forall i : 0 \leq i \leq n \quad ; \quad b_i = b'_i = 1 \quad (8.3)$$

Dans ces deux modèles de fautes, les valeurs des bits ciblés ne sont souvent pas connues avant l'attaque. L'effet d'une faute collée à une valeur fixe se montre quand la valeur initiale doit être réécrite à sa valeur opposée. A ce point, il peut créer un changement dans le comportement du système ou les résultats [Otto 2004].

- **Aléatoire** : Lorsque les changements de valeur sont aléatoires ; mais, la valeur d'au moins un des bits ciblés est changé. Autrement dit, le modèle de faute est *aléatoire*, si et seulement si :

$$\begin{aligned} \forall i : 0 \leq i \leq n \quad ; \quad b'_i \in \{0, 1\} \\ \text{and} \\ \exists ! j : 0 \leq j \leq n \quad ; \quad b'_j \neq b_j \end{aligned} \tag{8.4}$$

- **Set ou reset** : Quand les bits sont ciblés sont écrits à 1 ou effacés (écrits à 0) quelles que soit leur valeurs précédentes. Le modèle de faute est considérée comme *set*, si et seulement si :

$$\forall i : 0 \leq i \leq n \quad ; \quad b'_i = 1 \tag{8.5}$$

Contrairement, le modèle de faute est *reset*, si et seulement si :

$$\forall i : 0 \leq i \leq n \quad ; \quad b'_i = 0 \tag{8.6}$$

Parmi tous ces modèles, les fautes aléatoires sont souvent considérées comme les plus réalistes.

8.1.2 Attaques en fautes sur AES

AES est un algorithme de chiffrement qui traite les données du message clair par blocs de 128 bits. La clef secrète a une taille de 128, 196 ou 256 bits. Selon la longueur de la clef l'algorithme sera itéré 10, 12 ou 14 fois (rondes). AES est divisé en deux processus la diversification de la clef et le chiffrement des données [NIST 2001]. Nous montrons le schéma général d'AES-128 sur la Figure 8.3.

Pour la ronde initiale d'AES, l'algorithme utilise la clef secrète comme la clef de la ronde, mais pour chaque ronde suivante, la clef de la ronde est diversifiée à partir de sa précédente. Nous utilisons le préfixe « K » plus le numéro de la ronde pour indiquer une clef de ronde (par exemple : K_9 pour la clef de la ronde 9).

Le processus de chiffrement commence en prenant 16 octets (128 bits) du message en entrée, puis en les mettant dans une matrice de 4×4 octets.

Chaque ronde, sauf la première et la dernière comprend 4 étapes :

1. **SubBytes** échange la valeur de chaque octet de la matrice par une valeur correspondante dans une table prédéfinie appelée boîte de substitution.

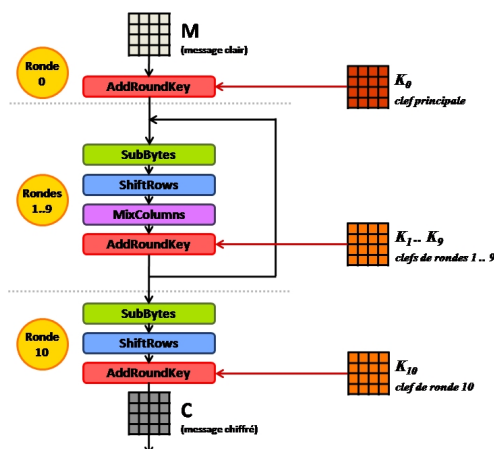


Figure 8.3: Algorithme AES-128.

2. **ShiftRows** consiste en des opérations de rotation vers gauche s'effectuant sur chaque ligne de la matrice.
3. **MixColumns** applique une transformation linéaire sur chaque élément en combinant celui-ci avec d'autres éléments de la même colonne par une addition avec différents multiples de valeurs 1, 2 ou 3 selon les règles spécifiques du corps fini $GF(2^8)$. Cette étape garantit la distribution de l'information de chaque octet sur 4 octets et donc l'augmentation de la sécurité des messages.
4. **AddRoundKey**, dernière étape de chaque ronde, correspond à une opération ou exclusif (xor) entre la valeur de chaque élément de la matrice et l'octet correspondant sur la clef de la ronde correspondante [NIST 2001].

8.1.3 Physique de l'injection de fautes par laser

Un laser est un dispositif émettant de la lumière (rayonnement électromagnétique) visible ou invisible amplifiée par une émission stimulée. Cette lumière introuvable dans la nature est monochromatique, unidirectionnelle et cohérente. Un faisceau de laser peut être produit dans un petit diamètre (jusqu'à $\varnothing 1\mu\text{m}$ au minimum) pour cibler un composant de très petite taille et altérer son fonctionnement sur une durée assez restreinte.

Un faisceau laser appliqué sur un circuit va perturber son fonctionnement normal. Par exemple sur une cellule de mémoire vive de type SRAM, en cas d'exposition au rayonnement du laser, il pourra modifier et/ou inverser le contenu des bits ciblés [Skorobogatov 2003].

En effet, selon l'architecture conventionnelle, chaque bit de SRAM est composé de deux inverseurs couplés pour stocker une valeur « 0 » ou « 1 » auxquels s'ajoutent

deux transistors qui contrôlent l'accès au contenu aux instants d'écriture ou de lecture.

Le contenu d'une cellule SRAM dépend du niveau logique en entrée et en sortie des inverseurs. Ils sont conçus et connectés de telle sorte que la cellule ne peut avoir qu'un des deux états stables « 0 » ou « 1 ». Si un faisceau laser arrive à toucher une zone sensible d'une SRAM, l'énergie qu'il y dépose pourra altérer son fonctionnement ou son contenu. Un faisceau laser crée le long de son parcours dans le silicium des paires électrons-trous par effets photoélectrique. Sous l'action d'un champ électrique, typiquement si elles sont localisées au niveau des drains des transistors bloqués, ces charges sont bloquées donnant naissance à un courant qui peut être suffisamment important pour changer le niveau logique d'un nœud de la SRAM. Cela amènera l'état de la cellule à basculer vers l'état inverse et changera le contenu de la cellule vers le contenu inverse [Darracq 2002].

Lors d'une attaque laser sur un circuit CMOS, plusieurs paramètres sont mis en jeu ; notamment : le diamètre du faisceau, la longueur d'onde, la quantité d'énergie émise et le temps d'exposition.

Il existe également des paramètres qui sont purement liés au circuit tels que l'instant d'injection (cycle d'horloge du processeur correspondant à l'instant d'exposition au laser), la technologie cible, et notamment la face d'entrée du faisceau car chacune des faces avant (active) ou arrière (substrat) d'un circuit a des caractéristiques différentes par rapport aux attaques laser.

8.2 Caractérisation sécuritaire

La caractérisation sécuritaire est une étape dans le flux classique de la conception des circuits intégrés. Son utilité est de vérifier la conformité des échantillons de puces aux spécifications de sécurité définies, dès le début de la phase de prototypage et même au cours de la production de masse.

La caractérisation sécuritaire se réfère à l'utilisation des techniques externes pour examiner la structure interne et les propriétés d'un circuit afin d'évaluer ses caractéristiques et sa résistance. Un circuit cryptographique, par exemple une carte à puce, doit avoir un niveau acceptable de sécurité afin d'atteindre les quatre objectifs de la sécurité et contre toute intervention non autorisée. La caractérisation sécuritaire utilise différents techniques d'analyse pour amplifier toute fuite de données ou toute vulnérabilité du circuit et aussi d'évaluer son inviolabilité.

8.2.1 Plateforme de caractérisation sécuritaire MircoPackS

Le laboratoire de caractérisation sécuritaire des circuits intégrés MicroPackS est une partie d'une plate-forme mutualisée du même nom entre les universitaires et industriels, située dans la région PACA, en France ¹. Son objectif principal est de

¹<http://www.arcsis.org>

fournir des équipements de caractérisation de sécurité pour évaluer des matériaux lors d'exécution de projets R & D, dès la phase de conception jusqu'à la phase de validation du produit.

Ce laboratoire est actuellement composé de cinq unités : Les bancs de caractérisation d'*injection de fautes laser*, d'*essais électriques*, d'*attaques SPA / DPA*, d'*attaques EMA* et des circuits *sans-contact*. La mise en place de la sixième unité pour un *banc laser d'injection de fautes laser avancé* est également prévue pour l'avenir proche.

Comme le sujet principal de cette thèse est l'injection de fautes par laser, nous nous concentrons uniquement sur les caractéristiques du banc laser actuel.

8.2.2 Caractéristiques du banc laser

Le banc d'injection de fautes laser est composé d'un émetteur de laser, un PC de commande, un oscilloscope et autres équipements nécessaires. L'émetteur de laser, présenté sur les figures 8.4 et 8.5, est équipé d'une source de laser YAG² avec trois différentes longueurs d'onde : 355 nm (ultraviolet), 532 nm (vert) et 1064 nm (infrarouge). La cible peut être fixée sur le *plateau motorisé pas à pas pour des microscopes verticaux* Prior Scientific, comme une table de positionnement X-Y programmable avec des pas de 0,1 μm .

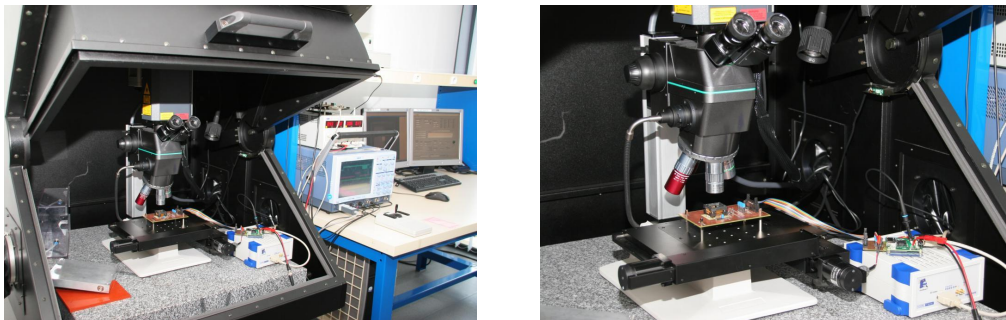


Figure 8.4: Plateforme laser MicroPackS.

Le faisceau a une forme nominale rectangulaire. Sa taille est programmable par l'ouverture d'obturateurs rectangulaires. Chacun des deux côtés parallèles d'obturateurs peut être ouvert dans une taille nominale entre 0 et 2500 μm . Comme le faisceau passe à travers d'une lentille de Mitutoyo, sa taille se réduit par le facteur de zoom de l'objectif et il perd une grande partie de son énergie.

La table XY, le lecteur de carte, l'émetteur laser et la carte de synchronisation FPGA du déclenchement de laser sont connectés par des liaisons RS-232 au PC de commande. Tous les paramètres sont contrôlés par une interface SMART-I³. Il s'agit

²Grenat d'Yttrium Aluminium ou $\text{Y}_3\text{Al}_5\text{O}_{12}$

³Secured Multi-characterisation Test Interface

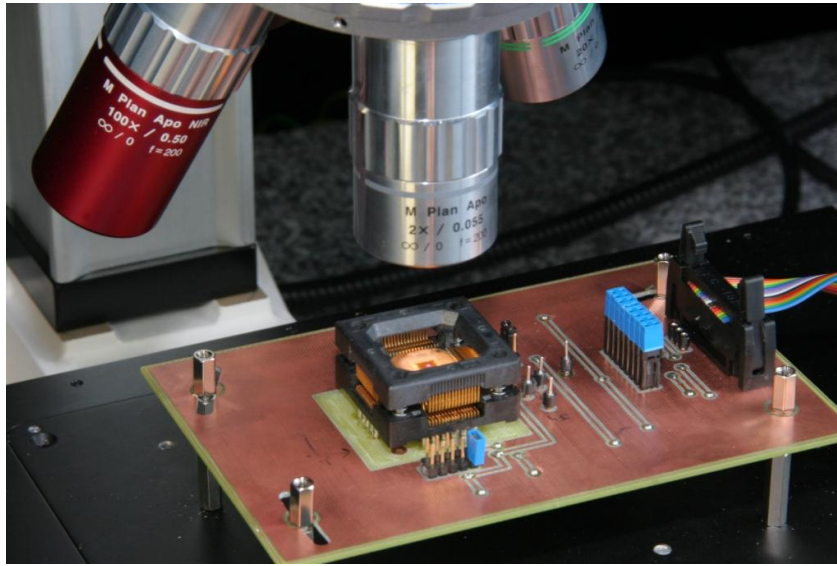


Figure 8.5: Plateforme laser et circuit test.

d'une interface graphique développée sous l'environnement LabVIEW pour les PCs de contrôle des différents bancs de caractérisation MicroPackS. SMARTI a la possibilité d'envoyer des commandes à des cartes à puce ou d'autres circuits sous test et obtenir et enregistrer des réponses et des courbes. La carte FPGA de déclenchement reçoit un signal d'activation à partir du lecteur et envoie un signal de déclenchement au laser après un retard défini sur l'interface SMART-I.

La durée d'une impulsion de laser est d'environ 5 ns. Un temps d'attente d'environ 200 μ s, avec une tolérance de 500 ns, est impliquée pour déclenchement interne de laser et commutation entre chaque commande de laser et son émission.

8.2.3 Préparation de circuit et des échantillons

Pour la plupart de nos essais de caractérisation dans cette thèse, nous avons utilisé une carte communicante aux normes carte à puce, fabriquée dans notre laboratoire. Elle est composée d'un microcontrôleur 8-bit 0,35 μ m 16 MHz avec une architecture RISC. Le microcontrôleur dispose de 128 kilo-octets mémoire flash pour programmation, 4 kilo-octets d'EEPROM et 4 kilo-octets de SRAM.

Le microcontrôleur est placé sur le circuit par l'intermédiaire d'un support du type ZIF qui permet de changer ses échantillons pour les essais différents. Une petite ouverture circulaire au centre de support de puce permet un accès physique à l'autre face de la puce pour des expériences laser sur la face arrière.

L'appareil fonctionne avec SOSSE ou le *Système d'exploitation simple pour l'éducation de carte à puce* [Bruestle 2002] pour simuler l'environnement de carte à puce.

8.2.3.1 Caractéristiques du circuit

SOSSE est un système d'exploitation open source et conforme aux commandes principales de la norme ISO/IEC 7816 pour les cartes à puce. Il a été développé comme un projet open source sous la licence publique GNU [Bruestle 2002]. SOSSE est principalement programmé en langage ANSI C.

SOSSE utilise le protocole T=0 pour la communication entre un lecteur de carte et une carte à puce [Bruestle 2002]. Le protocole T=0 est le protocole de communication, le plus largement utilisé pour des cartes à puce. Il a été normalisé sous ISO 7816-3 et 7816-4. Notre microcontrôleur communique à une fréquence de 3,59 MHz avec le lecteur de carte.

Pour effectuer nos essais, nous avons utilisé une version modifiée de SOSSE en s'y ajoutant dans notre laboratoire un ensemble de nouvelles commandes :

1. **Read_RAM** : Lecture d'une partie du contenu de SRAM.
2. **Write_RAM** : Écriture de la valeur d'octet donnée sur tous les octets de SRAM situés entre l'adresse 0x800500 et 0x8010FF.
3. **AES_Encrypt** : Envoi d'un texte clair afin de le chiffrer par l'AES embarqué.
4. **Read_Ciphertext** : Récupération du texte chiffré obtenu avec la commande précédente.

Dans l'implémentation de SOSSE, après chaque réinitialisation du circuit, toutes les variables sont copiées sur la SRAM. Dans notre version, la clé secrète K pour le chiffrement AES est incluse dans le code. Après chaque réinitialisation du circuit, toutes les clés de rondes sont calculées et stockées dans la SRAM. Le tableau de S-Box est également inclus dans le code du programme et après chaque réinitialisation est copié dans la SRAM. Le processus de chiffrement se réfère aux valeurs de clés de rondes et du tableau de S-Box mémorisées dans la SRAM au cours de chiffrement. La figure 8.6 montre un aperçu de l'implémentation de notre AES et de ses opérations principales.

Ainsi, dans le cadre des essais effectués pendant la préparation de cette thèse, le point d'entrée principal pour les attaques en fautes est le stockage de données sensibles dans la SRAM.

Ci-après, nous utilisons les termes “ notre puce”, “ circuit” ou “microcontrôleur” pour désigner notre microcontrôleur précité et utilisé pour la plupart des expériences, sauf quand un autre circuit est mentionné.

Pour les essais effectués avant l'implémentation des contre-mesures décrites dans la partie 8.5, notre circuit n'a eu aucune contre-mesure logicielle ou matérielle.

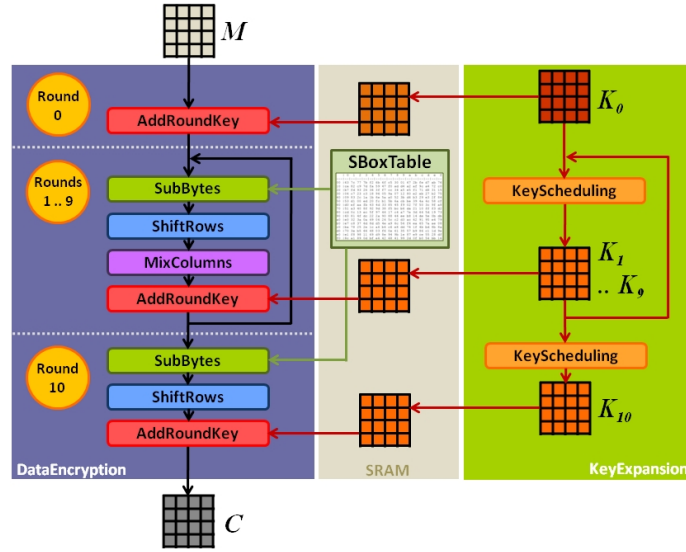


Figure 8.6: Un aperçu de l'implémentation de notre AES.

8.2.3.2 Décapsulation des échantillons

Avant de passer à l'étape de l'injection de fautes par laser, le circuit de cible doit être préparé. Les essais laser nécessitent la décapsulation préalable du circuit. Selon les équipements et les exigences d'attaque, une des faces avant ou arrière du circuit doit être décapsulée.

L'ouverture chimique est la méthode plus appropriée pour la décapsulation d'une puce par la face avant. Elle peut préserver l'intégrité et la fonctionnalité du circuit lorsqu'elle est effectuée de manière appropriée. La décapsulation chimique peut être effectuée manuellement par l'attaquant ou automatiquement ou semi-automatiquement par des machines décapsulation. L'emballage du circuit doit être retiré, tout en préservant l'intégrité du die, des plots, des fils et de la trame de fils d'interconnexions.

Pour une attaque laser sur la face arrière, la décapsulation mécanique est plus appropriée que les méthodes chimiques. Elle peut mieux entamer la distance entre la couche de silicium et les composants actifs. Pour réussir une attaque laser sur une puce décapsulée à la face arrière, la surface d'ouverture doit être parfaitement polie et amincie.

Pour nos essais, nous avons décapsulé plusieurs échantillons de notre circuit à la face avant ou arrière par des machines de décapsulation de la plateforme MicroPackS.

8.2.4 Caractérisation sécuritaire : Première cartographie de la susceptibilité d'injection de fautes

Avant de commencer nos essais de l'injection de fautes sur AES, une bonne connaissance de notre circuit et des effets des émissions laser sur son comportement étaient nécessaires.

Par le grossissement des images de la face avant des échantillons décapsulés, nous avons pu identifier des différentes composants de notre circuit, en fonction de leurs formes et compte tenu également des informations dans sa fiche technique.

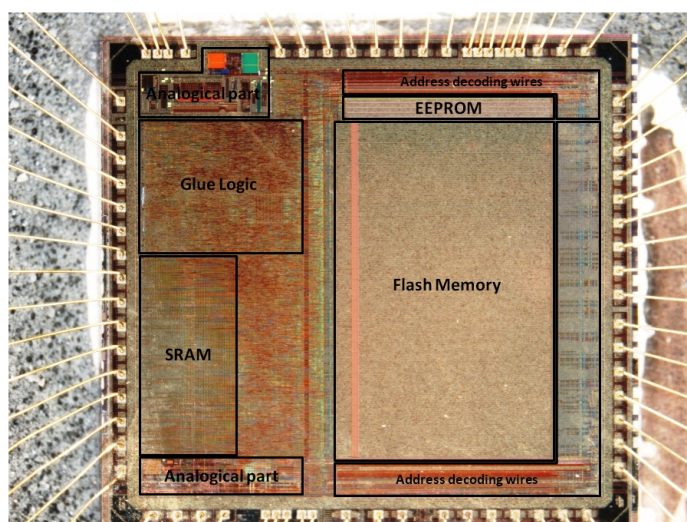


Figure 8.7: Cartographie du microcontrôleur cible.

La figure 8.7 montre la cartographie des principaux blocs de composants de la puce, dont la SRAM, la mémoire flash, l'EEPROM, les parties analogiques et logiques de notre circuit.

Puisque de nombreuses variables sont stockées dans la SRAM, des informations détaillées sur son organisation est nécessaire. Selon ce qu'on a dit sur les différents types de fautes, un tir laser sur la SRAM est susceptible de créer des bit-flips lorsque son énergie est inférieure au seuil de destruction. Donc, nous avons effectué des expériences préliminaires qui nous ont permis de mesurer les effets du laser sur notre circuit. Dans notre expérience, nous nous sommes interrogés sur la relation entre les paramètres du laser et le nombre de fautes injectées.

La taille du faisceau et le niveau d'énergie sont des paramètres impliqués sur le nombre de fautes. Pour des attaques menées sur la face arrière, une taille de faisceau plus grande et un niveau plus élevé d'énergie sont requis pour pouvoir injecter le même nombre de fautes en comparaison avec l'injection de faute sur la face avant. Cette différence entre les deux faces est expliquée par la perte d'énergie en passant la couche du silicium sur la face arrière.

8.2.4.1 Injection de fautes mono-bit et multi-bit

Le nombre de bits fautés sur chaque octet est un autre paramètre étudié au cours de nos expériences préliminaires. Nous avons constaté que lorsque la largeur du faisceau laser est plus petit que $\simeq 44 \mu\text{m}$, tous les octets fautés sur notre circuit contiennent un seul bit fauté. Avec l'augmentation de la taille du faisceaux laser, des octets fautés à deux ou plusieurs bits apparaissent. Cependant, nous ne pouvons pas créer plus de trois bits fautés sur chaque octet avec une largeur du faisceau, inférieure à $93 \mu\text{m}$. En outre, la fréquence des fautes à double-bits et triple-bits reste faible entre l'ensemble des octets fautés.

8.2.5 Exploration de notre circuit

En plus des expériences d'injection de fautes sur la SRAM, nous avons effectué une exploration complète de notre circuit par émission laser et l'analyse le comportement du circuit. Nous pouvons classer les effets de l'émission de laser sur notre circuit dans les quatre catégories suivantes :

1. Sans effet : Nous n'avons pas détecté aucune faute sur la SRAM, le chiffrement AES ou le comportement du circuit après l'émission laser.
2. Réponse fautée avec fonctionnement correct du circuit : Nous avons pu injecter des fautes par laser sur des valeurs stockées dans la SRAM ou sur les valeurs calculées ou communiquées. Toutefois, le circuit a maintenu son fonctionnement correct après l'émission laser.
3. Erreur temporaire du circuit : Parfois, l'émission laser a créé des comportements fautés temporaires. Dans ces cas, notre circuit n'a pas répondu à la première commande envoyée par le lecteur après l'émission laser. Mais, le circuit a retrouvé son bon fonctionnement rapidement.
4. Erreur permanente du circuit : Il y a également des cas de comportement fauté permanent du circuit. L'émission laser a engendré l'arrêt des réponses envoyées par notre circuit. En attendant plusieurs secondes ou minutes, le circuit n'a pas pu récupérer son fonctionnement correct. Dans ce cas, une *réinitialisation à froid* du circuit était nécessaire ; autrement dit, il fallait éteindre et rallumer la tension d'alimentation ou débrancher et de rebrancher le circuit du lecteur. Par conséquent, l'effet de ces fautes était permanent, mais pas destructif.

Pendant ces expériences préliminaires, nous avons obtenu une bonne connaissance des paramètres du laser pour injection de faute sur notre circuit. Nous avons ensuite utilisé de ces connaissances pour essayer des attaques classiques de DFA sur notre circuit.

8.3 Pratique de la DFA par laser sur AES

Nous avons présenté les principales techniques d'attaque en fautes. Parmi celles-ci, l'émission laser est une méthode d'injection de fautes avec une bonne localisation géographique et temporelle sur le circuit et ainsi, sur un crypto-système. Lors de nos essais préliminaires, nous avons prouvé la possibilité d'injection de fautes laser sur notre circuit. Nous avons examiné qu'il est possible de modifier certaines valeurs d'octets par émission laser sur la SRAM. Dans le contexte de nos essais de caractérisation sécuritaire des circuits, nous avons été amenés à nous interroger sur la faisabilité expérimentale de certains modèles d'attaque théoriques de DFA sur AES, dont [Piret 2003] et [Giraud 2005].

8.3.1 Problématiques de la pratique de la DFA par laser

Les attaques théoriques par DFA sur AES dépendent de modèles de fautes mono-octet ou mono-bit. Le diamètre minimal d'un faisceau laser ne peut pas être diminué à moins de $\varnothing 1\mu\text{m}$, pour des raisons de diffraction optique. En outre, par suite des progrès technologiques, le nombre de transistors placés sous un faisceau incident de $\varnothing 1\mu\text{m}$ augmente.

La figure 8.8 montre une comparaison entre des faisceaux laser de diamètres 1 et 10 μm et une cellule SRAM mise à l'échelle pour différentes technologies. Elle montre aussi qu'un faisceau laser de $\varnothing 1\mu\text{m}$ peut avoir une zone d'effet plus grande sur la puce par rapport à son niveau d'énergie. Ce faisceau minimal frappe plusieurs transistors sur les technologies les plus avancées et ne peut pas être physiquement limité à cibler un seul bit ou octet. Par ailleurs, un équipement laser de bonne précision qui produit un faisceau de $\varnothing 1\mu\text{m}$ est assez cher et n'est pas dans la mesure budgétaire de la plupart des attaquants. Donc, l'injection de fautes mono-bit/octet ou même sur quelques octets a besoin d'équipements bien précis et elle devient moins faisable par des équipements laser bon marché.

8.3.2 DFA mono-octet de Piret et Quisquater

Gilles Piret et Jean-Jacques Quisquater ont présenté dans [Piret 2003] une attaque DFA théorique sur AES. Cette attaque nécessite l'injection d'une faute sur un octet du texte chiffré temporaire entre la sortie `MixColumns` de la ronde antépénultième et l'entrée `MixColumns` de l'avant-dernière ronde pour réussir. La figure 8.9 montre la propagation de cette faute à partir de l'entrée du `MixColumns` de la ronde 9.

8.3.3 DFA mono-bit de Giraud

Christophe Giraud dans [Giraud 2005] a présenté deux attaques DFA sur AES. La première attaque exige des fautes mono-octet et la deuxième est une attaque mono-bit. Son attaque mono-bit nécessite l'injection d'une faute sur un seul bit d'un

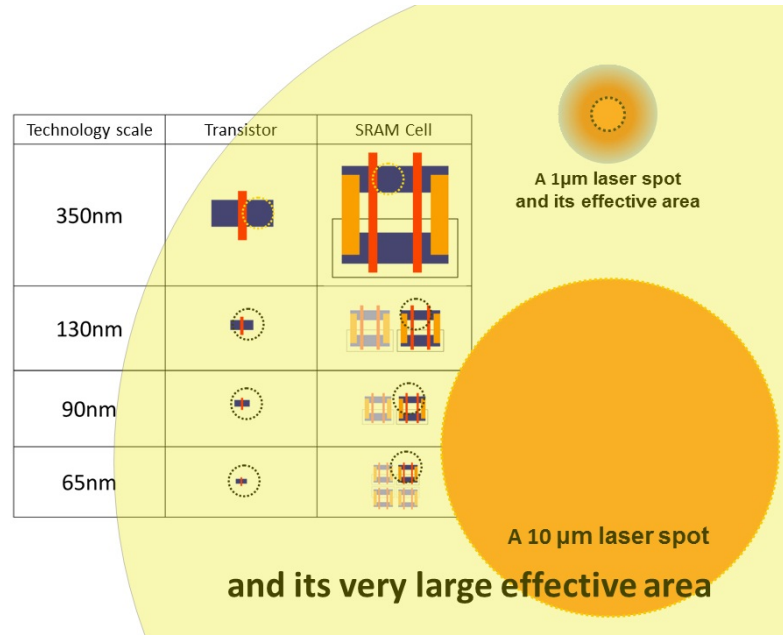


Figure 8.8: Mise en évidence de la taille de faisceaux laser d'1 μm et de 10 μm par rapport au progrès des technologies de fabrication.

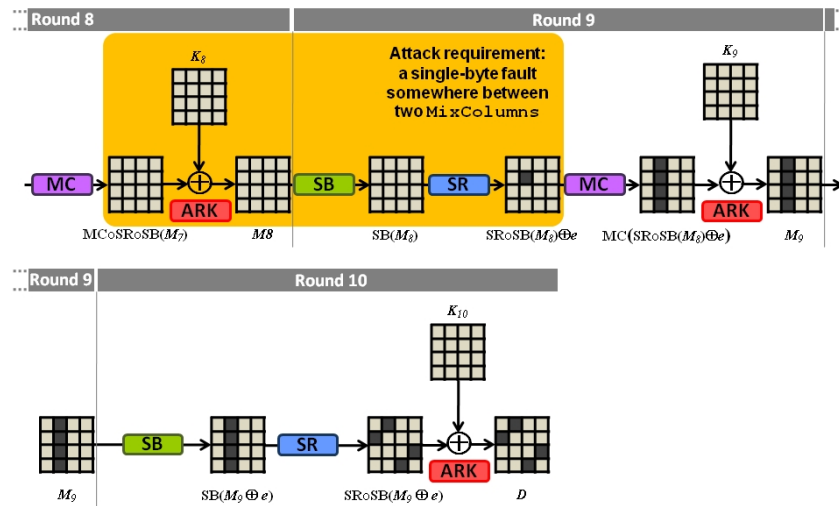


Figure 8.9: Propagation d'une faute mono-octet à l'entrée du MixColumns de la ronde 9.

octet spécifique du chiffré temporaire de l'avant-dernière ronde (M_9). La figure 8.10 montre l'exigence de cette attaque.

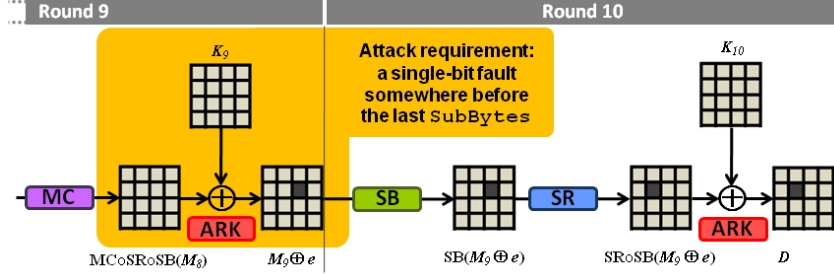


Figure 8.10: Attque DFA Giraud mono-bit.

Cette attaque n'est pas la plus performante, mais probablement la plus difficile à mettre en œuvre car elle est basée sur l'injection de faute sur un seul bit. Donc, la mise en place réussie de cette attaque, permettra à l'attaquant d'exercer successivement plusieurs autres modèles d'attaque en faute.

8.3.4 Résultats pratiques des DFA mono-octets et mono-bit

Dans nos essais, le faisceau de laser créait souvent des fautes supplémentaires par rapport aux modèles théoriques de DFA sur AES basés sur l'injection de faute mono-octet ou mono-bit. Nous avons réussi à exclure l'effet logique des fautes supplémentaires en jouant sur l'instant d'injection et sur le déplacement du spot laser. Notre solution est d'effectuer l'émission du laser et donc d'injecter des fautes dans l'intervalle de temps situé avant l'utilisation de l'octet ciblé dans les calculs de cryptographie, et après l'utilisation de tous les octets précédents qui sont éventuellement fautes. Donc, l'effet logique des fautes non-souhaitées sur les octets précédents est exclu, malgré le fait que ces fautes existent physiquement sur le circuit. Par conséquent, nous avons pu produire les fautes requises pour mener les attaques Piret-Quisquater et Giraud.

Dans une autre étape, par une méthode de classification de fautes, nous avons proposé la version étendue de l'attaque DFA Giraud mono-bit classique. Dans cette version, l'existence de plusieurs fautes mono-bit sur les octets de K_9 augmentait la vitesse d'obtention de la clefs et quelques fautes mono-bit sur K_{10} pouvaient être négligées par la classification. Ces résultats nous a amenés à des cas plus faisables concernant la diminution de la taille de fabrication des circuits et à identifier des menaces réalistes toujours existantes [Agoyan 2010a].

8.3.5 Une DFA multi-octets étendue

Après la mise en place réussie des attaques théoriques et leurs extensions par laser, nous avons étendue nos recherches à la découverte d'un nouveau modèle d'attaque

DFA, à son exploitation et à sa mise en place.

Nous avons étudié une attaque DFA qui exploite les fautes sur K_9 en présence de plusieurs fautes sur K_{10} . Dans ce cas, les octets fautés sur le chiffré résultent parfois de deux fautes injectées sur les octets correspondants de K_9 et de K_{10} .

Dans cette attaque, la valeur de chaque octet est trouvable en utilisant au minimum trois paires de chiffrés correct et fauté correspondants. Cette attaque semble être plus réaliste par rapport à la limite de diminution de la taille des faisceaux laser et l'augmentation de la densité des transistor dans les nouvelles technologies.

Les modèles classiques de DFA sont basés sur l'injection d'une faute mono-bit ou mono-octet sur une des dernières rondes. La différence remarquable de notre attaque est la liberté de l'injection de plusieurs fautes sur plusieurs bits ou octets, voire sur tous les octets des deux dernières rondes, après la sortie du `MixColumns` de l'avant-dernière ronde. Toutefois, cette attaque impose l'injection de fautes constantes lors de l'obtention des trois chiffrés fautés utilisés pour la cryptanalyse.

La conclusion de cette partie est un avertissement sur la faisabilité toujours actuelle des attaques DFA par laser. Nos essais ont montrés qu'il est toujours faisable de créer des fautes mono-octets ou mono-bits avec un faisceau de laser qui rencontre plusieurs octets sur le circuit quand l'injection de faute est bien maîtrisée et associée avec d'autres techniques. En outre, la découverte et la mise en place de nouvelles attaques, telle qu'une DFA multi-bit et multi-octet sur les deux rondes finales d'AES renforce cet aspect. Donc, l'existence des contre-mesures appropriées est nécessaire pour la conception des nouveaux circuits.

8.4 Attaques par modification de rondes

Après nos études sur les attaques DFA, nous avons continué nos recherches à la découverte de nouveaux modèles d'attaques en fautes, à leurs exploitations et à leurs mises en place. Les algorithmes de cryptographie symétrique sont souvent constitués de rondes répétées successivement. Les attaques en faute sur le compteur de rondes ou sur la référence du nombre de rondes ont été beaucoup moins étudiées par rapport aux attaques DFA.

Nous présentons brièvement les précédentes attaques sur la modification de rondes :

8.4.0.1 Attaque de H. Choukri et M. Tunstall

Selon la publication de H. Choukri et M. Tunstall [Choukri 2005], si l'attaquant arrive à changer le compteur de rondes (désormais CR) d'AES dès le commencement d'exécution de l'algorithme à la valeur finale (dans la publication à 0), le chiffré à la sortie sera le produit d'une seule ronde (en plus de la ronde initiale).

Avec cette attaque et selon l'algorithme d'AES, deux résultats différents sont envisageables :

1. Le chiffré est le produit d'exécution de la ronde initiale (R_0) plus d'une ronde intermédiaire ou normale d'AES (identique aux rondes $R_1..R_9$)
2. Le chiffré est le produit d'exécution de la ronde initiale (R_0) et de la ronde finale (R_{10}) d'AES

Dans ces deux cas, en utilisant deux paires de chiffrés fautés et corrects correspondants, la clef d'AES est retrouvable. L'attaque de H. Choukri et M. Tunstall [Choukri 2005] a été mise en place sur une implémentation naïve d'AES sans contre-mesure, embarqué sur un circuit PIC16F877. La méthode d'injection de faute était un pic de courant sur le V_{cc} , lors de l'exécution d'AES.

8.4.0.2 Attaque Y. Monnet et al.

La publication de Y. Monnet et al. [Monnet 2006] présente une autre attaque de réduction de rondes sur deux cryptoprocresseurs asynchrones implémentant l'algorithme de chiffrement DES. Le compteur de rondes utilisé dans ces deux cryptoprocresseurs a une implémentation multi-rail, telle que pour chacune des 16 rondes, une ligne séparée de signal existe. Une 17^{ème} ligne de signal a aussi été implémentée pour distinguer la fin d'exécution de l'algorithme. Ces lignes de signaux sont protégées par une alarme qui se déclenche à la détection de plus d'une seule ligne active en même temps.

Cette attaque a été menée par injection de faute par laser. Entre les deux cryptoprocresseurs asynchrones de DES, celui avec des contre-mesures avaient montré plus de résistance contre l'injection de faute pendant l'expérimentation. Toutefois, cette attaque a été mise en place avec réussite sur les deux circuits.

8.4.0.3 Attaque J. Park et al.

Le travail de J. Park et al. présenté dans la publication [Park 2011], il s'agit d'une attaque en faute par laser sur un AES embarqué dans un circuit ATmega128. Ils ont exploité une attaque de passage de la 1^{re} ronde à la ronde 10. Donc l'exécution fautée comprend les rondes : $R_0 - R_1 - R_{10}$, soit une ronde supplémentaire par rapport à l'attaque de H. Choukri et M. Tunstall [Choukri 2005] qui exécute $R_0 - R_{10}$. L'implémentation de l'algorithme AES dans cette attaque est celle proposée dans le document de référence [NIST 2001], annoncé par NIST.

Ces attaques précédentes sont basées sur la réduction du nombre de rondes. Pour nos essais, grâce à la précision obtenue lors de nos premiers travaux, nous avons développé plusieurs nouvelles attaques avec des solutions cryptanalytiques associées.

8.4.1 Modèles faisables d'attaque sur notre AES

En considérant l'implémentation de notre AES sous SOSSE, embarqué sur notre circuit, plusieurs possibilités sont envisageables. Pour cette raison, les résultats obtenus

lors de nos essais précédents pour l'injection de fautes par laser sur les faces avant et arrière de notre circuit ont été bien pris en compte.

Pour étudier la faisabilité de ces attaques, nous présentons les instructions de notre algorithme AES sur la figure 8.11 :

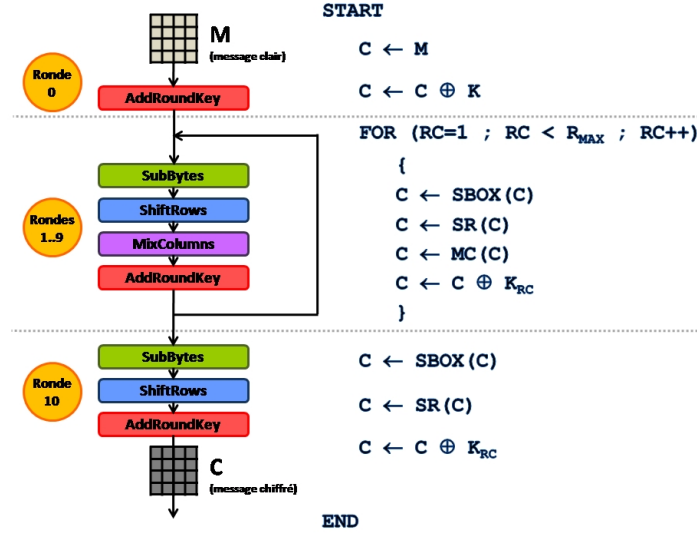


Figure 8.11: Les instructions de notre algorithme AES.

Dans notre implémentation, le compteur de ronde a été utilisé pour le comptage des rondes intermédiaires, désormais R_m (soit de la ronde 1 jusqu'à la ronde 9). Donc, même avec la suppression totale de l'exécution des rondes intermédiaires, les rondes initiale et finale seront exécutées.

Toutefois, l'indice de la clef utilisée lors d'ARK pour n'importe quelle ronde (même les rondes initiale et finale) est celui du compteur de rondes, CR . Donc, si la valeur du CR soit entre 1 et 10 ($1 \leq CR \leq 10$), le chiffré temporaire (l'état) entre dans une opération de xor avec K_{CR} . Mais, si la valeur du CR est supérieure à 10, l'algorithme recherchera les 16 octets stockés sur la mémoire à une adresse calculée par la même formule pour K_{CR} et utilisera les valeurs de ces 16 octets, même s'ils ne correspondent pas à une valeur valide de clef. Par conséquent, comme aucune clef d'AES-128 n'existe pas après K_{10} , le chiffré temporaire (état) entre dans une opération de xor avec une matrice de valeurs inconnues.

Le moment de l'injection de fautes pourra avoir des effets différents par rapport à l'instant d'exécution de l'algorithme ; même pour différents instants d'exécution de la même ronde. Donc, pour bien les caractériser, nous divisons ces différents instants d'exécution de l'algorithme :

1. FOR : Au début du cycle d'une ronde intermédiaire, entre l'affectation ou l'incrément de la valeur du CR et la comparaison de celui-ci avec sa valeur maximale.

2. LOOP : Après la comparaison du compteur de rondes avec la valeur maximale et avant ARK de la ronde actuelle.
3. ARK : Lors d'exécution d'ARK pour la ronde actuelle.
4. COMEBACK : Après l'exécution d'ARK et jusqu'à l'affectation de la nouvelle valeur à CR

Nous montrons ces différents instants de notre algorithme d'AES sur la figure 8.12.

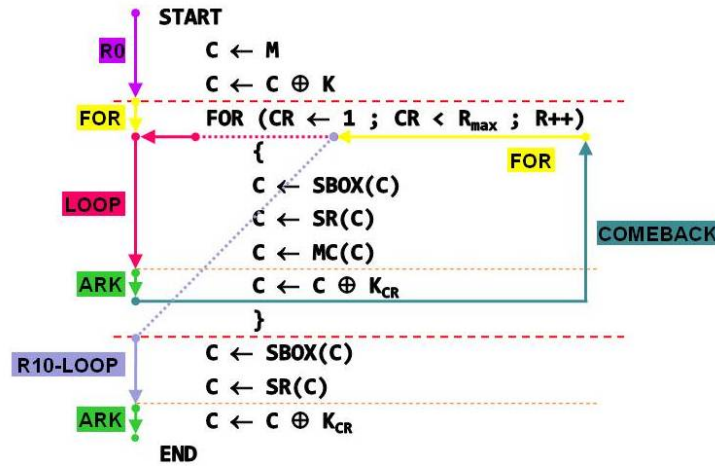


Figure 8.12: Différents moments d'exécution de notre algorithme AES par rapport aux utilisations du compteur de rondes.

8.4.2 Modèle d'attaques faisables sur notre AES

Pour bien étudier les possibilités d'attaque, nous révisons l'implémentation de notre algorithme, montré sur la figure 8.11.

Dans notre implémentation, le compteur de ronde est utilisé pour compter les étapes intermédiaires (R_m) (c'est-à-dire les rondes entre R_1 et R_9). La ronde initiale (R_0) et la ronde finale (R_{10}) sont implémentées séparément, comme le montre la figure 8.12. Donc, même dans le cas d'une suppression complète des rondes intermédiaires, la ronde initiale et la ronde finale seront exécutées.

Dans notre implémentation, le compteur de ronde (CR) et la référence du nombre de rondes ($R_{\max}$) d'AES sont stockées dans la SRAM du circuit. Ces deux valeurs peuvent être ciblées lors de l'injection de fautes par laser. Donc, deux principaux scénarios d'attaque sont envisageables :

Table 8.1: Conséquences de l'injection de faute sur le compteur de rondes lors des différents étapes d'exécution d'une ronde intermédiaire ou finale d'AES

étape	Conséquences
FOR	- changement du CR - augmentation ou réduction de nombre de rondes (éventuellement, réexécution ou suppression de quelques rondes)
LOOP	- changement du CR - changement du CR - exécution d'ARK d'une autre ronde (Si nouveau $CR > 10$, ARK avec valeurs inconnues) - augmentation ou réduction de nombre de rondes (éventuellement, réexécution ou suppression de quelques rondes)
ARK	- changement du CR - exécution partielle d'ARK d'une autre ronde (Si nouveau $CR > 10$, ARK partiel avec valeurs inconnues) - augmentation ou réduction de nombre de rondes (éventuellement, réexécution ou suppression de quelques rondes)
COMEBACK	- changement du CR - augmentation ou réduction de nombre de rondes (éventuellement, réexécution ou suppression de quelques rondes)

8.4.2.1 Attaque sur la valeur du compteur de rondes

Ce scénario d'attaque change le compteur de ronde lors d'exécution d'AES et par conséquent le numéro de la ronde en cours d'exécution. Selon le moment de l'injection de faute, différents changements sur l'exécution de l'algorithme pourront se produire. Les conséquences selon chaque étape de l'algorithme sont résumées dans le tableau 8.1.

8.4.2.2 Attaque sur la référence du nombre de rondes

Ce deuxième scénario d'attaque change la référence du nombre de rondes lors de l'exécution d'AES. Cette référence est consultée une seule fois par ronde et uniquement au début de chaque ronde intermédiaire et de la ronde finale d'AES. Pour la ronde finale, quelque soit la valeur du compteur de rondes, elle ne pourra jamais empêcher l'exécution de la ronde finale.

Selon le moment de l'injection de faute, différents changements sur l'exécution de l'algorithme pourront se produire. Les conséquences de ce type d'attaque sont résumées dans le tableau 8.2.

Table 8.2: Conséquences de l'injection de faute sur la référence du nombre de rondes lors des différents moments de l'exécution d'une ronde intermédiaire ou finale d'AES

Instant	Conséquences
FOR	- augmentation ou réduction du nombre de rondes (augmentation ou suppression de quelques rondes ; mais pas de réexécution) - prise en compte immédiate de la faute (dès la fin de FOR)
LOOP ARK	- augmentation ou réduction du nombre de rondes (augmentation ou suppression de quelques rondes ; mais pas de réexécution)
COMEBACK	- prise en compte de la faute dès le début de la prochaine ronde

Nos essais ont montré la faisabilité de ces types d'attaques. La conséquence de ces attaques sur le nombre de rondes exécuté et le chiffrement peut être un des cas suivants :

- Si $CR \oplus e < CR \Rightarrow$ *Augmentation de rondes* ou *exécution répétitive de plusieurs rondes*. Par exemple : si $CR=7$ et $e=2$; d'après $CR_{\text{fauté}} = CR \oplus e = 5$, l'exécution d'AES sera :

$$R_0..R_5-R_6-R_5-R_6-R_7..R_{10}$$

Les rondes 5 et 6 seront exécutées deux fois et le nombre total des rondes exécutées sera augmenté de 10 à 12.

- Si $CR \oplus e > CR$ et $CR \neq R_{\text{max}} - 1 \Rightarrow$ *Réduction de rondes*. Par exemple : si $CR=4$ et $e=2$; d'après $CR_{\text{fauté}} = CR \oplus e = 6$, l'exécution fautive d'AES sera :

$$R_0..R_3-R_6..R_{10}$$

Les rondes 4 et 5 seront supprimées et le nombre total des rondes exécutées sera diminué à 8.

- Si $CR \oplus e > CR$ et $CR = R_{\text{max}} - 1 \Rightarrow$ Sans aucun changement sur le nombre total des rondes exécutées ; mais avec des effets d'*Altération de rondes*. Dans ce cas, la transformation **AddRoundKey** de la ronde finale (et également, dans certaines conditions, celle de l'avant dernière ronde) seront exécutées avec des valeurs ne correspondant pas à une clef de ronde valide. Par exemple : si $CR=9$ et $e=2$; d'après $CR_{\text{fauté}} = CR \oplus e = 11$, la séquence de l'exécution d'AES sera :

$$R_0..R_8-R_{\text{m}=11}-R_{\text{f}=12}$$

Donc, le nombre total des rondes exécutées restera 10 ; mais, l'avant-dernière ronde et la ronde finale utiliseront des clefs de rondes invalides (K_{11} et K_{12}) lors de la transformation **AddRoundKey**.

Lors de nos études, nous avons identifiés une douzaines d'attaques sur une des cibles, soit CR , soit R_{max} , avec des cryptanalyses légères. Nous avons pu mettre en place 8 attaques parmi celles-ci. Nous les reportons brièvement dans le tableau 8.3 :

Table 8.3: Attaques exploitables sur CR ou R_{max} .

N°	cible	condition	exécution	textes	type
I-1	CR	$CR=1$ FOR	R_0-R_f	2	RR
I-2	CR	$CR=1$ LOOP	$R_0-R_m-R_f$	3	RR
I-3	CR	$CR=2$ ARK, COMEBACK ou $CR=3$ FOR	$R_0-R_1-R_f$	3	RR
I-4	CR	$CR=2$ LOOP accès à la mémoire requis	$R_0-R_1-R_m-R_f$	2	RR
I-5	CR	$CR=8$ ARK, COMEBACK si $e = \{1\}$	$R_0-R_1..R_8-R_f$	2	RR
I-6	CR	$CR=8$ ARK, COMEBACK si $e \notin \{1, 8\}$	$R_0-R_1..R_8-R_f$	3	RR
I-7	CR	$CR=9$ LOOP si $e \notin \{1, 8\}$	$R_0-R_1..R_8-R_m-R_f$	4	Alt.R
I-8	CR	$CR=9$ ARK, COMEBACK accès à la mémoire requis	$R_0-R_1..R_9-R_f$	1	Alt.R
II-1	R_{max}	Entre $CR=0$ et $CR=10$ FOR	$R_0..R_9-R_{m=10}$ $-R_{f=11}$	3	Aug.R
II-2	R_{max}	Entre $CR=0$ et $CR=10$ FOR accès à la mémoire requis	$R_0..R_9-4 \times R_{m=10..13}$ $-R_{f=14}$	1	Aug.R
II-3	R_{max}	Entre $CR=8$ LOOP et $CR=9$ FOR si $e \in \{2, 8\}$	$R_0..R_8-R_{f=9}$	3	RR

En plus des attaques présentées dans le tableau 8.3, nous avons étudié la possibilité de mise en place de deux attaques sur le CR , sur deux rondes consécutives et obtenu la cryptanalyse différentielle des chiffrés fautés.

En résumé, les attaques par modification de rondes forment une catégorie des attaques en faute. Elles comprennent les trois types d'attaques de réduction, d'augmentation et d'altération des rondes.

Nos essais montrent la possibilité de mise en place réussie de ces types d'attaques en absence des contre-mesures. Donc, l'implémentation des contre-mesures correspondantes est nécessaire contre les attaques par modification de rondes, en plus des attaques DFA.

8.5 Contre-mesures

Nos travaux précédents constituent un avertissement sur la faisabilité avérée des attaques par laser décrites dans la littérature scientifique. Cet avertissement est renforcé par l'introduction d'une nouvelle technique de DFA ; ainsi que d'une nouvelle technique d'attaque par modification d'algorithme : l'analyse de modification de rondes (qui est une extension des attaques par réduction de rondes). Donc, l'implémentation des contre-mesures appropriées est nécessaire pour la conception des nouveaux circuits. Pour ces raisons et afin de finaliser notre travail, nous avons proposé et évalué quelques contre-mesures classiques adaptées.

8.5.1 Contre-mesures contre l'analyse différentielle de fautes

Pour la protection de notre circuit, une contre-mesure idéale doit protéger les clefs de rondes de toute injection de fautes par laser. Il serait également très appréciable si cette contre-mesure pouvait garantir le circuit contre tout autre moyen d'attaques en fautes. Toutefois, une telle contre-mesure inspirerait d'intervenir au niveau de la conception du circuit. Donc, elle n'entre pas dans le cadre cette thèse. Pour cette raison, nous avons étudié des solutions embarquées contre ces attaques. En outre, toute solution matérielle peut être encore vulnérable par des méthodes à venir d'injection de fautes. Donc, même en utilisant des contre-mesures matérielles, des solutions embarquées pourront augmenter la sécurité.

Pour protéger les clefs de rondes contre toute injection de fautes ; nous avons mis en place une procédure de calcul inverse des clefs de rondes pour détecter tout changement de valeur lors de l'AddRoundKey et un contrôle de parité sur les valeurs de bits de chacune des clefs de rondes lors de son utilisation. En plus, un chiffrement parallèle à partir de la 6^{ème} ronde et comparaison des deux chiffrés à la fin du chiffrement permet de détecter toute injection de fautes exploitables lors des calculs. La figure 8.13 montre l'implémentation de ces contre-mesure contre les attaques DFA.

L'implémentation de ces contre-mesures augmente la sécurité de notre circuit contre les attaques DFA.

8.5.2 Contre-mesures contre l'analyse de modification de rondes

Lors de nos études sur des attaques RMA, nous avons pu cibler le compteur de rondes et également la référence du nombre de rondes. Ces attaques ont pu changer le nombre de rondes exécutées ; ainsi qu'altérer le chiffrement.

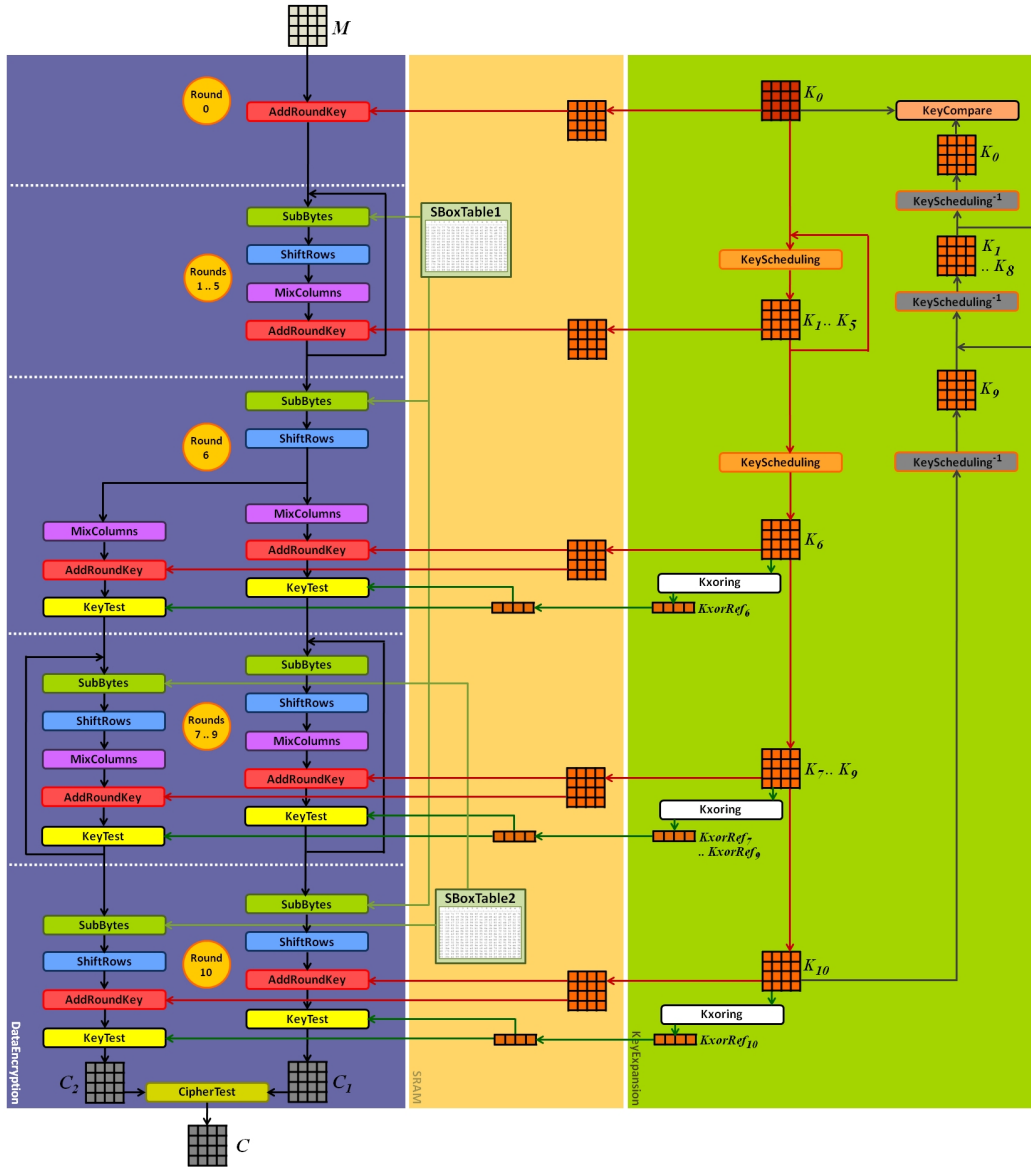


Figure 8.13: Processus parallèles de chiffrement et vérifications des clés dans notre implémentation d'AES.

Une contre-mesure contre les attaques RMA, doit protéger le circuit du changement de nombre de rondes et des altérations. Le déroulement des rondes d'AES est une solution pour couper la dépendance de l'exécution de l'algorithme vis-à-vis du compteur de rondes et à la référence du nombre de rondes.

Toutefois, d'autres points sensibles du circuit, comme le compteur du programme restent des cibles potentielles pour des attaques par modification de rondes. Donc, l'ajout de deux compteur de vérification de rondes, fonctionnant avec des opération de décalage de bit est une solution pour une double vérification et détection de tout

changement ou toute altération de rondes.

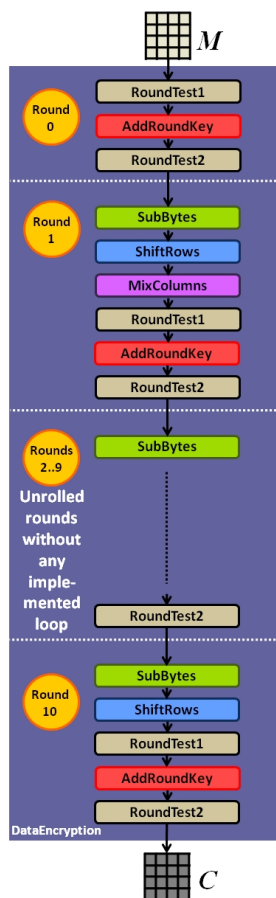


Figure 8.14: Implémentation de l’AES déroulé avec deux vérifications à chaque ronde.

La figure 8.14 montre notre contre-mesure contre les attaques RMA.

8.5.3 Une contre-mesure combinée contre DFA et RMA

Les deux contre-mesures contre DFA et RMA pourront être combinées afin de protéger notre AES de ces deux types d’attaques. La figure 8.15 montre la combinaison de ces deux contre-mesures.

Nos propositions de contre-mesures classiques embarquées pourront protéger notre AES contre les attaques DFA et RMA par laser. Comme la menace des attaques en fautes est bien présente sur un circuit non-protégé, l’utilisation et la mise en place de ces contre-mesures sont obligatoires afin de protéger les circuits cryptographiques.

Les contre-mesures présentées ont pour rôle de préserver notre AES des attaques décrites précédemment. Elles pourront être adaptées à la protection d’autres implémentations.

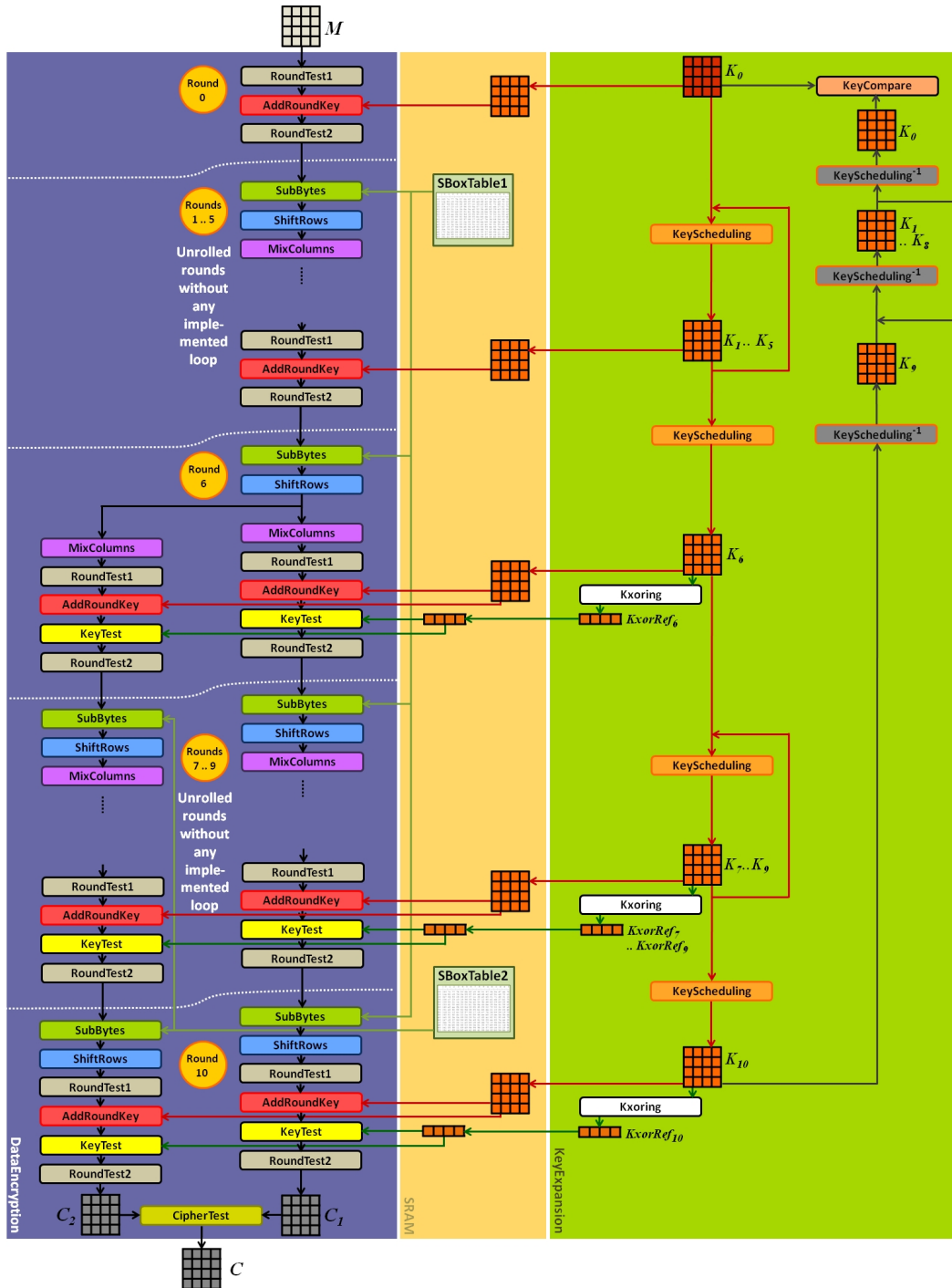


Figure 8.15: Contre-mesure combinée pour des attaques DFA et RMA dans notre implémentation d'AES.

Toutefois, elles prennent une partie importante des ressources matérielles du circuit et augmentent le temps de chiffrement. Cela est un compromis inévitable afin de renforcer la protection des informations sensibles dans un crypto-système implémentant AES.

8.6 Autres perspectives de sécurité

Étant intéressés aux recherches en sécurité, nous avons contribué à quelques sujets connexes ; dont une parade contre les attaques d'analyse de consommation sur les cartes à puce.

Dans le cadre d'un autre travail de recherche, nous avons également évalué la résistance d'une contre-mesure aux attaques en fautes par modification de la température.

Ici, nous présentons brièvement notre parade contre les attaques d'analyse de consommation sur les cartes à puce.

8.6.1 Une contre-mesure efficace contre les attaques d'analyse de consommation

Quand la ligne d'alimentation d'une carte à puce est reliée à son contact externe, aucune contre-mesure ne peut pas cacher entièrement les variations de consommation d'énergie du regard de l'attaquant. La seule solution est de séparer la ligne d'alimentation de son contact. Un moyen de satisfaire cette exigence est de fournir une alimentation électrique séparée dans la structure de la carte. Cette alimentation intégrée aura toujours besoin d'une source d'alimentation externe lui fournissant son énergie par le contact de la carte. Cet ensemble doit masquer entièrement la consommation de la puce à l'extérieure de la carte. Il est assez difficile de trouver une telle source d'énergie électrique qui pourrait tenir dans la structure de la carte à puce et fournir suffisamment d'énergie pour l'alimentation de la puce.

Un ensemble constitué d'une source de lumière et d'une cellule photovoltaïque pourrait être un choix adéquat, s'il satisfait les restrictions de taille, de puissance et les exigences de commutation rapides pour une carte à puce. Parmi les sources de lumière possibles, une *diode électroluminescente* (LED) ou une LED *organique* (OLED) pourraient être des bons choix correspondants à ces critères.

Après avoir sélectionné différents modèles de LED, le seul modèle de panneau OLED disponible et quelques cellules photovoltaïques ayant de bonnes performances, nous avons examiné leur efficacité pour cet essai. Nous avons obtenu les meilleurs résultats avec le panneau OLED Osram Orbeos CDW-031 [Semiconductors 2009] et la cellule photovoltaïque Sanyo Amorton AM-8801 [Sanyo 2008]. Donc, nous les avons utilisés pour fabriquer notre contre-mesure.

Comme montre la figure 8.16, cette contre-mesure est basée sur l'utilisation d'un sandwich constitué d'une diode électroluminescente organique (OLED) de quelques

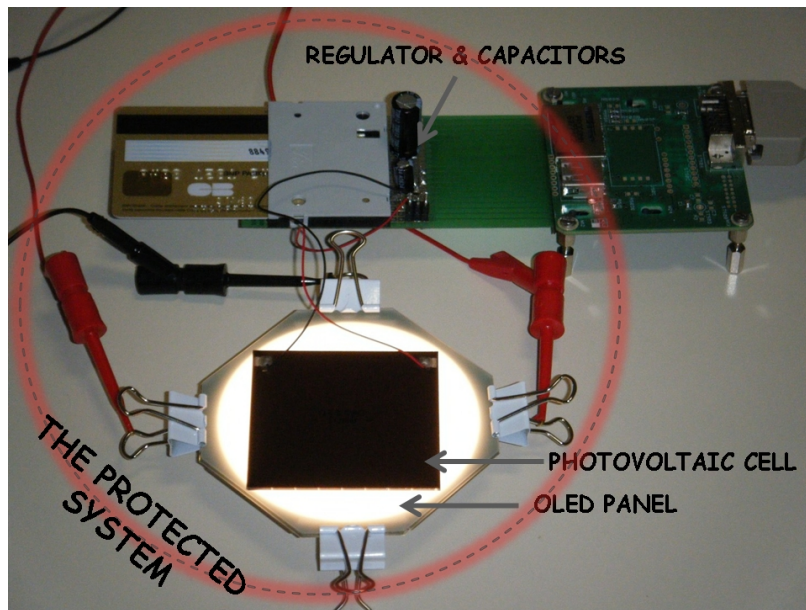


Figure 8.16: Système protégé par la contre-mesure.

centaines de nanomètres d'épaisseur et d'une cellule photovoltaïque ayant une épaisseur de quelques dizaines de micromètres afin de rendre la consommation de la carte indépendante des données manipulées. Nous avons également utilisé un régulateur de tension Infineon TLE 4264 afin de maintenir la stabilité de la tension à 5V. En plus, nous avons utilisé deux condensateurs électrolytiques de 100 μF et 10 μF , respectivement à l'entrée et la sortie du régulateur.

La partie inférieure de la figure 8.17 montre le schéma électronique de la contre-mesure avec séparation des masses. Cet ensemble pourra être implémenté à l'intérieur d'une futur génération de carte à puce et la protéger grâce à la constance de la consommation de la cellule OLED.

Normalement, nous aurions dû relier toutes les masses comme la partie supérieure de la figure 8.17 le montre. Mais, cette configuration pourrait donner la possibilité d'une nouvelle attaque potentielle, par l'analyse de la consommation sur le contact de masse sur la carte. Pour éviter ce problème, nous avons utilisé une solution afin de ne pas connecter les deux lignes de la masse. Comme le montre la figure 8.17, nous avons considéré une connexion paire torsadée entre les contacts d'I/O et d'horloge de la carte et du lecteur. Le principe de cette solution est comme la mise en œuvre de deux lignes de données D+ et D- dans un Bus universel en série (USB) pour établir une connexion de transfert direct de données entre deux composants, sans l'utilisation d'une ligne de masse commune.

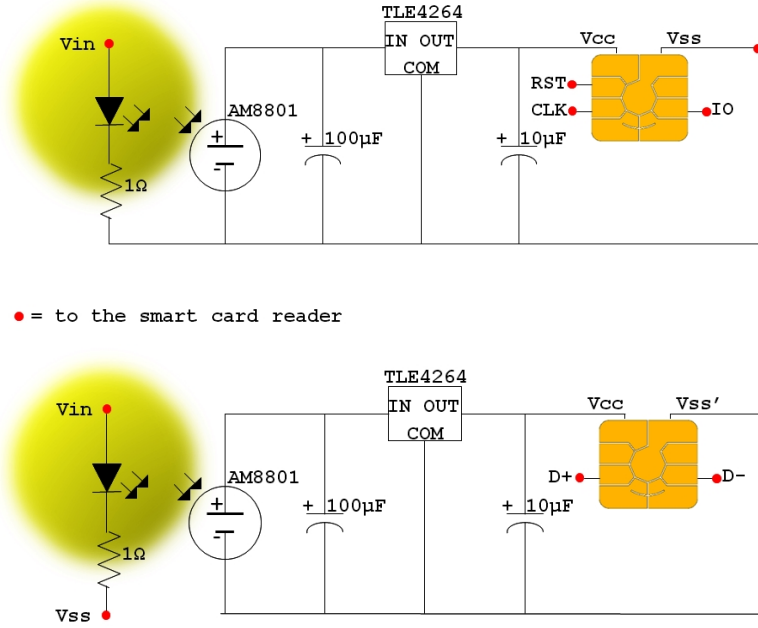


Figure 8.17: Schéma électronique de la contre-mesure sans séparation des masses (haut) et avec séparation des masses (bas).

8.7 Conclusions et perspectives

L'objectif de ce travail de thèse était l'identification et l'étude des menaces liées à l'injection de fautes par laser sur les systèmes cryptographiques.

Dans un premier temps, nous avons effectué nos recherches sur la validation des modèles de fautes théoriques par laser. Nous avons utilisé un microcontrôleur implémentant un algorithme de cryptographie AES. L'analyse différentielle de fautes (DFA) est une méthode d'attaque en fautes pour découvrir une clef secrète, par comparaison des chiffrés corrects et fautés correspondants. Dans nos essais, le faisceau de laser créait souvent des fautes supplémentaires par rapport aux modèles théoriques de DFA sur AES (ces modèles requièrent l'injection de fautes mono-octets ou mono-bits).

En effet, avec le progrès des technologies de fabrication, la taille des composants des circuits intégrés ne cesse de diminuer. Par ailleurs, la taille minimale d'un faisceau laser est de l'ordre d'environ $1\ \mu\text{m}$ de diamètre à cause de la diffraction des ondes. L'obtention de cette taille minimale nécessite l'utilisation d'optiques coûteuses accessibles à seulement un nombre réduit d'attaquants.

Ces facteurs conjugués semblent rendre de plus en plus difficile l'injection de fautes mono-octet et mono-bit et peuvent conduire à s'interroger sur la faisabilité expérimentale des attaques théoriques.

A contrario, nos travaux ont consisté à mettre en évidence le caractère toujours

pertinent de ces modèles d'attaques et d'éviter ainsi tout risque de sous-estimation de la menace.

En effet, nous avons réussi à exclure l'effet logique des fautes supplémentaires par un jeu précis sur l'instant et le lieu d'injection. Malgré le fait que ces fautes existent physiquement sur le circuit, la seule faute prise en compte lors du calcul cryptographique est celle requise par l'attaque mise en œuvre. En outre, par une méthode de classification des fautes, nous avons proposé une version étendue de DFA Giraud-bit, une attaque mono-bit classique sur AES.

Ensuite, nous avons étendue nos recherches à la découverte de nouveaux modèles d'attaques en fautes, à leurs exploitations et à leurs mises en pratique. Les algorithmes de cryptographie symétrique sont souvent constitués de rondes répétées successivement. Les attaques en faute sur le compteur de rondes ou sur la référence du nombre de rondes ont été beaucoup moins étudiées par rapport aux attaques DFA. Grâce à la précision obtenue lors de nos premiers travaux, nous avons développé une quinzaine de nouvelles attaques avec les solutions cryptanalytiques associées. Elles visent deux cibles : le compteur de ronde et la référence du nombre de rondes d'AES. Une dizaine de ces attaques est basée sur des scénarios simples, comprenant une attaque, à un moment précis de l'exécution de l'algorithme, sur une des deux cibles susdites.

En conclusion, les travaux précédents constituent un avertissement sur la faisabilité avérée des attaques par laser décrites dans la littérature scientifique. Nos essais ont témoigné de la faisabilité toujours existante de la mise en place des attaques mono-octets ou mono-bits avec un faisceau de laser qui rencontre plusieurs octets sur le circuit quand ils sont bien maîtrisés et associés avec d'autres techniques. Donc, l'implémentation de contre-mesures appropriées est nécessaire pour la conception des nouveaux circuits. Pour ces raisons et afin d'achever notre travail, nous avons proposé et évalué quelques contre-mesures classiques adaptées.

Par ailleurs, étant intéressés aux recherches en sécurité, nous avons contribué à quelques sujets connexes ; dont une parade contre les attaques d'analyse de consommation sur les cartes à puce. Cette contre-mesure est basée sur l'utilisation d'un sandwich constitué d'une diode électroluminescente organique (OLED) de quelques centaines de nanomètres d'épaisseur et d'une cellule photovoltaïque ayant une épaisseur de quelques dizaines de micromètres afin de rendre la consommation de la carte indépendante des données manipulées. Cet ensemble pourra être implémenté à l'intérieure d'une futur génération de carte à puce et la protéger grâce à la constance de la consommation de la cellule OLED. Dans le cadre d'un autre travail de recherche, nous avons également évalué la résistance d'une contre-mesure aux attaques en fautes par modification de la température.

Nos perspectives pour continuer ces recherches sont :

- **Recherche sur d'autres mécanismes de l'injection de fautes sur microcontrôleur :** L'étude de l'injection de fautes sur l'unité logique arithmétique (UAL), le bus et l'opération d'expansion de clef.

- **Étude de la vulnérabilité des mémoires flash :** Dont des recherches sur la modification du programme chargé par des fautes et également sur l'injection de fautes sur les valeurs initiales.
- **Implémentation et validation de contremesures :** Afin de pouvoir protéger des circuits de l'injection de fautes et de toute fuite des informations sensibles par fautes.

Nous espérons de poursuivre ces perspectives de recherche pour nos prochains travaux à la suite de cette thèse.

Appendix RMA Results

In this appendix, we describe all the consequences of Round Modification Analysis (RMA) attacks in the context of our circuit. We report only single-bit fault injection cases either into the round counter (RC) or into the reference of total round number (R_{max}). These cases also include the attacks that are not exploitable by light cryptanalysis solutions. The purpose of these tables is to present an overview of different single-bit attacks. For studying the feasibility of multi-bit fault injection, many other cases must be reviewed.

A.1 Attacks on the Round Counter Value

We show here all the possible algorithm executions when a single-bit fault is injected into the round counter value. Expanding these attacks to multi-bit models creates larger exploitation possibilities.

Table A.1: All the single-bit attacks of scenario I.

Round counter	Attack moment	Model of injected fault							
		0x01	0x02	0x04	0x08	0x10	0x20	0x40	0x80
$RC = 1$		$RC \rightarrow 0$	$RC \rightarrow 3$	$RC \rightarrow 5$	$RC \rightarrow 9$	$RC \rightarrow 17$	$RC \rightarrow 33$	$RC \rightarrow 65$	$RC \rightarrow 129$
	FOR	$N R \rightarrow 11$ $R_0 - R_{1m} - R_1 .. R_{10}$ m=0	$N R \rightarrow 8$ $R_0 - R_3 .. R_{10}$	$N R \rightarrow 6$ $R_0 - R_5 .. R_{10}$	$N R \rightarrow 2$ $R_0 - R_9 .. R_{10}$	$N R \rightarrow 1$ $R_0 - R_f$ f=17	$N R \rightarrow 1$ $R_0 - R_f$ f=33	$N R \rightarrow 1$ $R_0 - R_f$ f=65	$N R \rightarrow 1$ $R_0 - R_f$ f=129
	LOOP	$N R \rightarrow 11$ $R_0 - R_m - R_1 .. R_{10}$ m=0	$N R \rightarrow 8$ $R_0 - R_3 .. R_{10}$	$N R \rightarrow 6$ $R_0 - R_5 .. R_{10}$	$N R \rightarrow 2$ $R_0 - R_9 .. R_{10}$	$N R \rightarrow 2$ $R_0 - R_m - R_f$ m=17 f=18	$N R \rightarrow 2$ $R_0 - R_m - R_f$ m=33 f=34	$N R \rightarrow 2$ $R_0 - R_m - R_f$ m=65 f=66	$N R \rightarrow 2$ $R_0 - R_m - R_f$ m=129 f=130
	ARK COMEBACK	$N R \rightarrow 11$ $R_0 - R_1 - R_1 .. R_{10}$	$N R \rightarrow 8$ $R_0 - R_1 - R_4 .. R_{10}$	$N R \rightarrow 6$ $R_0 - R_1 - R_6 .. R_{10}$	$N R \rightarrow 2$ $R_0 - R_1 - R_{10}$	$N R \rightarrow 2$ $R_0 - R_1 - R_f$ f=18	$N R \rightarrow 2$ $R_0 - R_1 - R_f$ f=34	$N R \rightarrow 2$ $R_0 - R_1 - R_f$ f=66	$N R \rightarrow 2$ $R_0 - R_1 - R_f$ f=130
$RC = 2$		$RC \rightarrow 3$	$RC \rightarrow 0$	$RC \rightarrow 6$	$RC \rightarrow 10$	$RC \rightarrow 18$	$RC \rightarrow 34$	$RC \rightarrow 66$	$RC \rightarrow 130$
	FOR	$N R \rightarrow 9$ $R_0 - R_1 - R_3 .. R_{10}$	$N R \rightarrow 12$ $R_0 - R_1 - R_m - R_1 .. R_{10}$ m=0	$N R \rightarrow 6$ $R_0 - R_1 - R_6 .. R_{10}$	$N R \rightarrow 2$ $R_0 - R_1 - R_{10}$	$N R \rightarrow 2$ $R_0 - R_1 - R_f$ f=18	$N R \rightarrow 2$ $R_0 - R_1 - R_f$ f=34	$N R \rightarrow 2$ $R_0 - R_1 - R_f$ f=66	$N R \rightarrow 2$ $R_0 - R_1 - R_f$ f=130
	LOOP	$N R \rightarrow 9$ $R_0 - R_1 - R_3 .. R_{10}$	$N R \rightarrow 12$ $R_0 - R_1 - R_m - R_1 .. R_{10}$ m=0	$N R \rightarrow 6$ $R_0 - R_1 - R_6 .. R_{10}$	$N R \rightarrow 3$ $R_0 - R_1 - R_m - R_f$ m=10 f=11	$N R \rightarrow 3$ $R_0 - R_1 - R_m - R_f$ m=18 f=19	$N R \rightarrow 3$ $R_0 - R_1 - R_m - R_f$ m=34 f=35	$N R \rightarrow 3$ $R_0 - R_1 - R_m - R_f$ m=66 f=67	$N R \rightarrow 3$ $R_0 - R_1 - R_m - R_f$ m=130 f=131
	ARK COMEBACK	$N R \rightarrow 9$ $R_0 ... R_2 - R_4 .. R_{10}$	$N R \rightarrow 12$ $R_0 ... R_2 - R_1 .. R_{10}$	$N R \rightarrow 7$ $R_0 ... R_2 - R_6 .. R_{10}$	$N R \rightarrow 3$ $R_0 ... R_2 - R_f$ f=11	$N R \rightarrow 3$ $R_0 ... R_2 - R_f$ f=19	$N R \rightarrow 3$ $R_0 ... R_2 - R_f$ f=35	$N R \rightarrow 3$ $R_0 ... R_2 - R_f$ f=67	$N R \rightarrow 3$ $R_0 ... R_2 - R_f$ f=131

Round counter	Attack moment	Model of injected fault							
		0x01	0x02	0x04	0x08	0x10	0x20	0x40	0x80
$RC' = 3$		$RC \rightarrow 2$	$RC \rightarrow 1$	$RC \rightarrow 7$	$RC \rightarrow 11$	$RC \rightarrow 19$	$RC \rightarrow 35$	$RC \rightarrow 67$	$RC \rightarrow 131$
	FOR	$N R \rightarrow 11$ $R_0..R_2..R_2..R_{10}$	$N R \rightarrow 12$ $R_0..R_2..R_1..R_{10}$	$N R \rightarrow 6$ $R_0..R_2..R_7..R_{10}$	$N R \rightarrow 3$ $R_0..R_2..R_f$ f=11	$N R \rightarrow 3$ $R_0..R_2..R_f$ f=19	$N R \rightarrow 3$ $R_0..R_2..R_f$ f=35	$N R \rightarrow 3$ $R_0..R_2..R_f$ f=67	$N R \rightarrow 3$ $R_0..R_2..R_f$ f=131
	LOOP	$N R \rightarrow 11$ $R_0..R_2..R_2..R_{10}$	$N R \rightarrow 12$ $R_0..R_2..R_1..R_{10}$	$N R \rightarrow 6$ $R_0..R_2..R_7..R_{10}$	$N R \rightarrow 4$ $R_0..R_2..R_m..R_f$ m=11 f=12	$N R \rightarrow 4$ $R_0..R_2..R_m..R_f$ m=19 f=20	$N R \rightarrow 4$ $R_0..R_2..R_m..R_f$ m=35 f=36	$N R \rightarrow 4$ $R_0..R_2..R_m..R_f$ m=67 f=68	$N R \rightarrow 4$ $R_0..R_2..R_m..R_f$ m=131 f=132
	ARK COMEBACK	$N R \rightarrow 11$ $R_0..R_3..R_3..R_{10}$	$N R \rightarrow 12$ $R_0..R_3..R_2..R_{10}$	$N R \rightarrow 6$ $R_0..R_3..R_8..R_{10}$	$N R \rightarrow 4$ $R_0..R_3..R_f$ f=12	$N R \rightarrow 4$ $R_0..R_3..R_f$ f=20	$N R \rightarrow 4$ $R_0..R_3..R_f$ f=36	$N R \rightarrow 4$ $R_0..R_3..R_f$ f=68	$N R \rightarrow 4$ $R_0..R_3..R_f$ f=132
$RC = 4$		$RC' \rightarrow 5$	$RC \rightarrow 6$	$RC \rightarrow 0$	$RC \rightarrow 12$	$RC \rightarrow 20$	$RC \rightarrow 36$	$RC \rightarrow 68$	$RC \rightarrow 132$
	FOR	$N R \rightarrow 9$ $R_0..R_3..R_5..R_{10}$	$N R \rightarrow 8$ $R_0..R_3..R_6..R_{10}$	$N R \rightarrow 14$ $R_0..R_3..R_m..R_1..R_{10}$ m=0	$N R \rightarrow 4$ $R_0..R_3..R_f$ f=12	$N R \rightarrow 4$ $R_0..R_3..R_f$ f=20	$N R \rightarrow 4$ $R_0..R_3..R_f$ f=36	$N R \rightarrow 4$ $R_0..R_3..R_f$ f=68	$N R \rightarrow 4$ $R_0..R_3..R_f$ f=132
	LOOP	$N R \rightarrow 9$ $R_0..R_3..R_5..R_{10}$	$N R \rightarrow 8$ $R_0..R_3..R_6..R_{10}$	$N R \rightarrow 14$ $R_0..R_3..R_m..R_1..R_{10}$ m=0	$N R \rightarrow 5$ $R_0..R_3..R_m..R_f$ m=12 f=13	$N R \rightarrow 5$ $R_0..R_3..R_m..R_f$ m=20 f=21	$N R \rightarrow 5$ $R_0..R_3..R_m..R_f$ m=36 f=37	$N R \rightarrow 5$ $R_0..R_3..R_m..R_f$ m=68 f=69	$N R \rightarrow 5$ $R_0..R_3..R_m..R_f$ m=132 f=133
	ARK COMEBACK	$N R \rightarrow 9$ $R_0..R_4..R_6..R_{10}$	$N R \rightarrow 8$ $R_0..R_4..R_7..R_{10}$	$N R \rightarrow 14$ $R_0..R_4..R_1..R_{10}$	$N R \rightarrow 5$ $R_0..R_4..R_f$ f=13	$N R \rightarrow 5$ $R_0..R_4..R_f$ f=21	$N R \rightarrow 5$ $R_0..R_4..R_f$ f=37	$N R \rightarrow 5$ $R_0..R_4..R_f$ f=69	$N R \rightarrow 5$ $R_0..R_4..R_f$ f=133

Table continued on next page

Table continued on next page

Table A.1 – Table continued from previous page

Round counter	Attack moment	Model of injected fault							
		0x01	0x02	0x04	0x08	0x10	0x20	0x40	0x80
RC = 5		RC→4	RC→7	RC→1	RC→13	RC→21	RC→37	RC→69	RC→133
	FOR	$NR \rightarrow 11$ $R_0..R_4-R_4..R_{10}$	$NR \rightarrow 8$ $R_0..R_4-R_7..R_{10}$	$NR \rightarrow 14$ $R_0..R_4-R_1..R_{10}$	$NR \rightarrow 5$ $R_0..R_4-R_f$ $f=13$	$NR \rightarrow 5$ $R_0..R_4-R_f$ $f=21$	$NR \rightarrow 5$ $R_0..R_4-R_f$ $f=37$	$NR \rightarrow 5$ $R_0..R_4-R_f$ $f=69$	$NR \rightarrow 5$ $R_0..R_4-R_f$ $f=133$
	LOOP	$NR \rightarrow 11$ $R_0..R_4-R_4..R_{10}$	$NR \rightarrow 8$ $R_0..R_4-R_7..R_{10}$	$NR \rightarrow 14$ $R_0..R_4-R_1..R_{10}$	$NR \rightarrow 6$ $R_0..R_4-R_{1m}-R_f$ $m=13$ $f=14$	$NR \rightarrow 6$ $R_0..R_4-R_{1m}-R_f$ $m=21$ $f=22$	$NR \rightarrow 6$ $R_0..R_4-R_{1m}-R_f$ $m=37$ $f=38$	$NR \rightarrow 6$ $R_0..R_4-R_{1m}-R_f$ $m=69$ $f=70$	$NR \rightarrow 6$ $R_0..R_4-R_{1m}-R_f$ $m=133$ $f=134$
	ARK COMEBACK	$NR \rightarrow 11$ $R_0..R_5-R_5..R_{10}$	$NR \rightarrow 8$ $R_0..R_5-R_8..R_{10}$	$NR \rightarrow 14$ $R_0..R_5-R_2..R_{10}$	$NR \rightarrow 6$ $R_0..R_5-R_f$ $f=14$	$NR \rightarrow 6$ $R_0..R_5-R_f$ $f=22$	$NR \rightarrow 6$ $R_0..R_5-R_f$ $f=38$	$NR \rightarrow 6$ $R_0..R_5-R_f$ $f=70$	$NR \rightarrow 6$ $R_0..R_5-R_f$ $f=134$
RC = 6		RC→7	RC→4	RC→2	RC→14	RC→22	RC→38	RC→70	RC→134
	FOR	$NR \rightarrow 9$ $R_0..R_5-R_7..R_{10}$	$NR \rightarrow 12$ $R_0..R_5-R_4..R_{10}$	$NR \rightarrow 14$ $R_0..R_5-R_2..R_{10}$	$NR \rightarrow 6$ $R_0..R_5-R_f$ $f=14$	$NR \rightarrow 6$ $R_0..R_5-R_f$ $f=22$	$NR \rightarrow 6$ $R_0..R_5-R_f$ $f=38$	$NR \rightarrow 6$ $R_0..R_5-R_f$ $f=70$	$NR \rightarrow 6$ $R_0..R_5-R_f$ $f=134$
	LOOP	$NR \rightarrow 9$ $R_0..R_5-R_7..R_{10}$	$NR \rightarrow 12$ $R_0..R_5-R_4..R_{10}$	$NR \rightarrow 14$ $R_0..R_5-R_2..R_{10}$	$NR \rightarrow 7$ $R_0..R_5-R_{1m}-R_f$ $m=14$ $f=15$	$NR \rightarrow 7$ $R_0..R_5-R_{1m}-R_f$ $m=22$ $f=23$	$NR \rightarrow 7$ $R_0..R_5-R_{1m}-R_f$ $m=38$ $f=39$	$NR \rightarrow 7$ $R_0..R_5-R_{1m}-R_f$ $m=70$ $f=71$	$NR \rightarrow 7$ $R_0..R_5-R_{1m}-R_f$ $m=134$ $f=135$
	ARK COMEBACK	$NR \rightarrow 9$ $R_0..R_6-R_8..R_{10}$	$NR \rightarrow 12$ $R_0..R_6-R_5..R_{10}$	$NR \rightarrow 14$ $R_0..R_6-R_3..R_{10}$	$NR \rightarrow 7$ $R_0..R_6-R_f$ $f=15$	$NR \rightarrow 7$ $R_0..R_6-R_f$ $f=23$	$NR \rightarrow 7$ $R_0..R_6-R_f$ $f=39$	$NR \rightarrow 7$ $R_0..R_6-R_f$ $f=71$	$NR \rightarrow 7$ $R_0..R_6-R_f$ $f=135$
Table continued on next page									

Table A.1 – Table continued from previous page

Round counter	Attack moment	Model of injected fault							
		0x01	0x02	0x04	0x08	0x10	0x20	0x40	0x80
$RC = 7$		$RC \rightarrow 6$	$RC \rightarrow 5$	$RC \rightarrow 3$	$RC \rightarrow 15$	$RC \rightarrow 23$	$RC \rightarrow 39$	$RC \rightarrow 71$	$RC \rightarrow 135$
	FOR	$N R \rightarrow 11$ $R_0 \dots R_6 \rightarrow R_6 \dots R_{10}$	$N R \rightarrow 12$ $R_0 \dots R_6 \rightarrow R_5 \dots R_{10}$	$N R \rightarrow 14$ $R_0 \dots R_6 \rightarrow R_3 \dots R_{10}$	$N R \rightarrow 7$ $R_0 \dots R_6 \rightarrow R_f$ $f=15$	$N R \rightarrow 7$ $R_0 \dots R_6 \rightarrow R_f$ $f=23$	$N R \rightarrow 7$ $R_0 \dots R_6 \rightarrow R_f$ $f=39$	$N R \rightarrow 7$ $R_0 \dots R_6 \rightarrow R_f$ $f=71$	$N R \rightarrow 7$ $R_0 \dots R_6 \rightarrow R_f$ $f=135$
	LOOP	$N R \rightarrow 11$ $R_0 \dots R_6 \rightarrow R_6 \dots R_{10}$	$N R \rightarrow 12$ $R_0 \dots R_6 \rightarrow R_5 \dots R_{10}$	$N R \rightarrow 14$ $R_0 \dots R_6 \rightarrow R_3 \dots R_{10}$	$N R \rightarrow 8$ $R_0 \dots R_6 \rightarrow R_m \rightarrow R_f$ $m=15$ $f=16$	$N R \rightarrow 8$ $R_0 \dots R_6 \rightarrow R_m \rightarrow R_f$ $m=23$ $f=24$	$N R \rightarrow 8$ $R_0 \dots R_6 \rightarrow R_m \rightarrow R_f$ $m=39$ $f=40$	$N R \rightarrow 8$ $R_0 \dots R_6 \rightarrow R_m \rightarrow R_f$ $m=71$ $f=72$	$N R \rightarrow 8$ $R_0 \dots R_6 \rightarrow R_m \rightarrow R_f$ $m=135$ $f=136$
	ARK COMEBACK	$N R \rightarrow 11$ $R_0 \dots R_7 \rightarrow R_7 \dots R_{10}$	$N R \rightarrow 12$ $R_0 \dots R_7 \rightarrow R_6 \dots R_{10}$	$N R \rightarrow 14$ $R_0 \dots R_7 \rightarrow R_4 \dots R_{10}$	$N R \rightarrow 8$ $R_0 \dots R_7 \rightarrow R_f$ $f=16$	$N R \rightarrow 8$ $R_0 \dots R_7 \rightarrow R_f$ $f=24$	$N R \rightarrow 8$ $R_0 \dots R_7 \rightarrow R_f$ $f=40$	$N R \rightarrow 8$ $R_0 \dots R_7 \rightarrow R_f$ $f=72$	$N R \rightarrow 8$ $R_0 \dots R_7 \rightarrow R_f$ $f=136$
$RC = 8$		$RC \rightarrow 9$	$RC \rightarrow 10$	$RC \rightarrow 12$	$RC \rightarrow 0$	$RC \rightarrow 24$	$RC \rightarrow 40$	$RC \rightarrow 72$	$RC \rightarrow 136$
	FOR	$N R \rightarrow 9$ $R_0 \dots R_7 \rightarrow R_9 \dots R_{10}$	$N R \rightarrow 8$ $R_0 \dots R_7 \rightarrow R_{10}$	$N R \rightarrow 8$ $R_0 \dots R_7 \rightarrow R_f$ $f=12$	$N R \rightarrow 18$ $R_0 \dots R_7 \rightarrow R_m \rightarrow R_1 \dots R_{10}$ $m=0$	$N R \rightarrow 8$ $R_0 \dots R_7 \rightarrow R_f$ $f=24$	$N R \rightarrow 8$ $R_0 \dots R_7 \rightarrow R_f$ $f=40$	$N R \rightarrow 8$ $R_0 \dots R_7 \rightarrow R_f$ $f=72$	$N R \rightarrow 8$ $R_0 \dots R_7 \rightarrow R_f$ $f=136$
	LOOP	$N R \rightarrow 9$ $R_0 \dots R_7 \rightarrow R_9 \dots R_{10}$	$N R \rightarrow 9$ $R_0 \dots R_7 \rightarrow R_m \rightarrow R_f$ $m=10$ $f=11$	$N R \rightarrow 9$ $R_0 \dots R_7 \rightarrow R_m \rightarrow R_f$ $m=12$ $f=13$	$N R \rightarrow 18$ $R_0 \dots R_7 \rightarrow R_m \rightarrow R_1 \dots R_{10}$ $m=0$	$N R \rightarrow 9$ $R_0 \dots R_7 \rightarrow R_m \rightarrow R_f$ $m=24$ $f=25$	$N R \rightarrow 9$ $R_0 \dots R_7 \rightarrow R_m \rightarrow R_f$ $m=40$ $f=41$	$N R \rightarrow 9$ $R_0 \dots R_7 \rightarrow R_m \rightarrow R_f$ $m=72$ $f=73$	$N R \rightarrow 9$ $R_0 \dots R_7 \rightarrow R_m \rightarrow R_f$ $m=136$ $f=137$
	ARK COMEBACK	$N R \rightarrow 9$ $R_0 \dots R_8 \rightarrow R_{10}$	$N R \rightarrow 9$ $R_0 \dots R_8 \rightarrow R_f$ $f=11$	$N R \rightarrow 9$ $R_0 \dots R_8 \rightarrow R_f$ $f=13$	$N R \rightarrow 18$ $R_0 \dots R_8 \rightarrow R_1 \dots R_{10}$	$N R \rightarrow 9$ $R_0 \dots R_8 \rightarrow R_f$ $f=25$	$N R \rightarrow 9$ $R_0 \dots R_8 \rightarrow R_f$ $f=41$	$N R \rightarrow 9$ $R_0 \dots R_8 \rightarrow R_f$ $f=73$	$N R \rightarrow 9$ $R_0 \dots R_8 \rightarrow R_f$ $f=137$

Table A.1 – Table continued from previous page

Round counter	Attack moment	Model of injected fault							
		0x01	0x02	0x04	0x08	0x10	0x20	0x40	0x80
<i>RC</i> = 9		<i>RC</i>→8	<i>RC</i>→11	<i>RC</i>→13	<i>RC</i>→1	<i>RC</i>→25	<i>RC</i>→41	<i>RC</i>→73	<i>RC</i>→137
	FOR	<i>NR</i> →11 <i>R</i> ₀ .. <i>R</i> ₈ - <i>R</i> ₈ .. <i>R</i> ₁₀ <i>f</i> =11	<i>NR</i> →9 <i>R</i> ₀ .. <i>R</i> ₈ - <i>R</i> _f <i>f</i> =13	<i>NR</i> →9 <i>R</i> ₀ .. <i>R</i> ₈ - <i>R</i> _f	<i>NR</i> →18 <i>R</i> ₀ .. <i>R</i> ₈ - <i>R</i> ₁ .. <i>R</i> ₁₀ <i>R</i> ₀ .. <i>R</i> ₈ - <i>R</i> ₁ .. <i>R</i> ₁₀	<i>NR</i> →9 <i>R</i> ₀ .. <i>R</i> ₈ - <i>R</i> _f <i>f</i> =25	<i>NR</i> →9 <i>R</i> ₀ .. <i>R</i> ₈ - <i>R</i> _f <i>f</i> =41	<i>NR</i> →9 <i>R</i> ₀ .. <i>R</i> ₈ - <i>R</i> _f <i>f</i> =73	<i>NR</i> →9 <i>R</i> ₀ .. <i>R</i> ₈ - <i>R</i> _f <i>f</i> =137
	LOOP	<i>NR</i> →11 <i>R</i> ₀ .. <i>R</i> ₈ - <i>R</i> ₈ .. <i>R</i> ₁₀	<i>NR</i> →10 <i>R</i> ₀ .. <i>R</i> ₈ - <i>R</i> _m - <i>R</i> _f <i>m</i> =11 <i>f</i> =12	<i>NR</i> →10 <i>R</i> ₀ .. <i>R</i> ₈ - <i>R</i> _m - <i>R</i> _f <i>m</i> =13 <i>f</i> =14	<i>NR</i> →18 <i>R</i> ₀ .. <i>R</i> ₈ - <i>R</i> ₁ .. <i>R</i> ₁₀	<i>NR</i> →10 <i>R</i> ₀ .. <i>R</i> ₈ - <i>R</i> _m - <i>R</i> _f <i>m</i> =25 <i>f</i> =26	<i>NR</i> →10 <i>R</i> ₀ .. <i>R</i> ₈ - <i>R</i> _m - <i>R</i> _f <i>m</i> =41 <i>f</i> =42	<i>NR</i> →10 <i>R</i> ₀ .. <i>R</i> ₈ - <i>R</i> _m - <i>R</i> _f <i>m</i> =73 <i>f</i> =74	<i>NR</i> →10 <i>R</i> ₀ .. <i>R</i> ₈ - <i>R</i> _m - <i>R</i> _f <i>m</i> =137 <i>f</i> =138
	ARK COMEBACK	<i>NR</i> →11 <i>R</i> ₀ .. <i>R</i> ₉ - <i>R</i> ₉ .. <i>R</i> ₁₀	<i>NR</i> →10 <i>R</i> ₀ .. <i>R</i> ₉ - <i>R</i> _f <i>f</i> =12	<i>NR</i> →10 <i>R</i> ₀ .. <i>R</i> ₉ - <i>R</i> _f <i>f</i> =14	<i>NR</i> →18 <i>R</i> ₀ .. <i>R</i> ₉ - <i>R</i> ₂ .. <i>R</i> ₁₀	<i>NR</i> →10 <i>R</i> ₀ .. <i>R</i> ₉ - <i>R</i> _f <i>f</i> =26	<i>NR</i> →10 <i>R</i> ₀ .. <i>R</i> ₉ - <i>R</i> _f <i>f</i> =42	<i>NR</i> →10 <i>R</i> ₀ .. <i>R</i> ₉ - <i>R</i> _f <i>f</i> =74	<i>NR</i> →10 <i>R</i> ₀ .. <i>R</i> ₉ - <i>R</i> _f <i>f</i> =138
<i>RC</i> = 10		<i>RC</i>→11	<i>RC</i>→8	<i>RC</i>→14	<i>RC</i>→2	<i>RC</i>→26	<i>RC</i>→42	<i>RC</i>→74	<i>RC</i>→138
	FOR	<i>NR</i> →10 <i>R</i> ₀ .. <i>R</i> ₉ - <i>R</i> _f <i>f</i> =11	<i>NR</i> →12 <i>R</i> ₀ .. <i>R</i> ₉ - <i>R</i> ₈ .. <i>R</i> ₁₀	<i>NR</i> →10 <i>R</i> ₀ .. <i>R</i> ₉ - <i>R</i> _f <i>f</i> =14	<i>NR</i> →18 <i>R</i> ₀ .. <i>R</i> ₉ - <i>R</i> ₂ .. <i>R</i> ₁₀	<i>NR</i> →10 <i>R</i> ₀ .. <i>R</i> ₉ - <i>R</i> _f <i>f</i> =26	<i>NR</i> →10 <i>R</i> ₀ .. <i>R</i> ₉ - <i>R</i> _f <i>f</i> =42	<i>NR</i> →10 <i>R</i> ₀ .. <i>R</i> ₉ - <i>R</i> _f <i>f</i> =74	<i>NR</i> →10 <i>R</i> ₀ .. <i>R</i> ₉ - <i>R</i> _f <i>f</i> =138
	LOOP	<i>NR</i> →10 <i>R</i> ₀ .. <i>R</i> ₉ - <i>R</i> _f <i>f</i> =11	<i>NR</i> →10 <i>R</i> ₀ .. <i>R</i> ₉ - <i>R</i> _f <i>f</i> =8	<i>NR</i> →10 <i>R</i> ₀ .. <i>R</i> ₉ - <i>R</i> _f <i>f</i> =14	<i>NR</i> →10 <i>R</i> ₀ .. <i>R</i> ₉ - <i>R</i> _f <i>f</i> =2	<i>NR</i> →10 <i>R</i> ₀ .. <i>R</i> ₉ - <i>R</i> _f <i>f</i> =26	<i>NR</i> →10 <i>R</i> ₀ .. <i>R</i> ₉ - <i>R</i> _f <i>f</i> =42	<i>NR</i> →10 <i>R</i> ₀ .. <i>R</i> ₉ - <i>R</i> _f <i>f</i> =74	<i>NR</i> →10 <i>R</i> ₀ .. <i>R</i> ₉ - <i>R</i> _f <i>f</i> =138

A.2 Attacks on the Round Number Reference

We show here all the possible algorithm executions when the round number reference is the target of a single-bit fault injection. Expanding these attacks to multi-bit models creates larger exploitation possibilities.

Table A.2 – Table continued from previous page

Round counter	Attack moment	Model of injected fault							
		0x01	0x02	0x04	0x08	0x10	0x20	0x40	0x80
		$R_{max} \rightarrow 11$ $R_{max} \rightarrow 8$ $R_{max} \rightarrow 14$ $R_{max} \rightarrow 2$ $R_{max} \rightarrow 26$ $R_{max} \rightarrow 42$ $R_{max} \rightarrow 74$ $R_{max} \rightarrow 138$							
$RC = 3$									
	FOR	$NR \rightarrow 11$ $R_0 \dots R_9 \times R_m \text{-} R_f$ $m=10$ $f=11$	$NR \rightarrow 8$ $R_0 \dots R_7 \text{-} R_f$ $f=8$	$NR \rightarrow 14$ $R_0 \dots R_9 \text{-} 4 \times R_m \text{-} R_f$ $m=\{10, 11, \dots 13\}$ $f=14$	$NR \rightarrow 3$ $R_0 \dots R_2 \text{-} R_f$ $f=3$	$NR \rightarrow 26$ $R_0 \dots R_9 \text{-} 16 \times R_m \text{-} R_f$ $m=\{10, 11, \dots 25\}$ $f=26$	$NR \rightarrow 42$ $R_0 \dots R_9 \text{-} 32 \times R_m \text{-} R_f$ $m=\{10, 11, \dots 41\}$ $f=42$	$NR \rightarrow 74$ $R_0 \dots R_9 \text{-} 64 \times R_m \text{-} R_f$ $m=\{10, 11, \dots 73\}$ $f=74$	$NR \rightarrow 138$ $R_0 \dots R_9 \text{-} 128 \times R_m \text{-} R_f$ $m=\{10, 11, \dots 137\}$ $f=138$
	LOOP ARK COMEBACK	$NR \rightarrow 11$ $R_0 \dots R_9 \times R_m \text{-} R_f$ $m=10$ $f=11$	$NR \rightarrow 8$ $R_0 \dots R_7 \text{-} R_f$ $f=8$	$NR \rightarrow 14$ $R_0 \dots R_9 \text{-} 4 \times R_m \text{-} R_f$ $m=\{10, 11, \dots 13\}$ $f=14$	$NR \rightarrow 4$ $R_0 \dots R_3 \text{-} R_f$ $f=4$	$NR \rightarrow 26$ $R_0 \dots R_9 \text{-} 16 \times R_m \text{-} R_f$ $m=\{10, 11, \dots 25\}$ $f=26$	$NR \rightarrow 42$ $R_0 \dots R_9 \text{-} 32 \times R_m \text{-} R_f$ $m=\{10, 11, \dots 41\}$ $f=42$	$NR \rightarrow 74$ $R_0 \dots R_9 \text{-} 64 \times R_m \text{-} R_f$ $m=\{10, 11, \dots 73\}$ $f=74$	$NR \rightarrow 138$ $R_0 \dots R_9 \text{-} 128 \times R_m \text{-} R_f$ $m=\{10, 11, \dots 137\}$ $f=138$
$RC = 4$									
	FOR	$NR \rightarrow 11$ $R_0 \dots R_9 \times R_m \text{-} R_f$ $m=10$ $f=11$	$NR \rightarrow 8$ $R_0 \dots R_7 \text{-} R_f$ $f=8$	$NR \rightarrow 14$ $R_0 \dots R_9 \text{-} 4 \times R_m \text{-} R_f$ $m=\{10, 11, \dots 13\}$ $f=14$	$NR \rightarrow 4$ $R_0 \dots R_3 \text{-} R_f$ $f=4$	$NR \rightarrow 26$ $R_0 \dots R_9 \text{-} 16 \times R_m \text{-} R_f$ $m=\{10, 11, \dots 25\}$ $f=26$	$NR \rightarrow 42$ $R_0 \dots R_9 \text{-} 32 \times R_m \text{-} R_f$ $m=\{10, 11, \dots 41\}$ $f=42$	$NR \rightarrow 74$ $R_0 \dots R_9 \text{-} 64 \times R_m \text{-} R_f$ $m=\{10, 11, \dots 73\}$ $f=74$	$NR \rightarrow 138$ $R_0 \dots R_9 \text{-} 128 \times R_m \text{-} R_f$ $m=\{10, 11, \dots 137\}$ $f=138$
	LOOP ARK COMEBACK	$NR \rightarrow 11$ $R_0 \dots R_9 \times R_m \text{-} R_f$ $m=10$ $f=11$	$NR \rightarrow 8$ $R_0 \dots R_7 \text{-} R_f$ $f=8$	$NR \rightarrow 14$ $R_0 \dots R_9 \text{-} 4 \times R_m \text{-} R_f$ $m=\{10, 11, \dots 13\}$ $f=14$	$NR \rightarrow 5$ $R_0 \dots R_4 \text{-} R_f$ $f=5$	$NR \rightarrow 26$ $R_0 \dots R_9 \text{-} 16 \times R_m \text{-} R_f$ $m=\{10, 11, \dots 25\}$ $f=26$	$NR \rightarrow 42$ $R_0 \dots R_9 \text{-} 32 \times R_m \text{-} R_f$ $m=\{10, 11, \dots 41\}$ $f=42$	$NR \rightarrow 74$ $R_0 \dots R_9 \text{-} 64 \times R_m \text{-} R_f$ $m=\{10, 11, \dots 73\}$ $f=74$	$NR \rightarrow 138$ $R_0 \dots R_9 \text{-} 128 \times R_m \text{-} R_f$ $m=\{10, 11, \dots 137\}$ $f=138$

Table continued on next page

Table A.2 – Table continued from previous page

Round counter	Attack moment	Model of injected fault											
		0x01	0x02	0x04	0x08	0x10	0x20	0x40	0x80				
RC = 5		$R_{max} \rightarrow 11$		$R_{max} \rightarrow 8$	$R_{max} \rightarrow 14$		$R_{max} \rightarrow 2$	$R_{max} \rightarrow 26$		$R_{max} \rightarrow 42$	$R_{max} \rightarrow 74$		$R_{max} \rightarrow 138$
	FOR	$NR \rightarrow 11$ $R_0 \dots R_9 \text{--} R_m \text{--} R_f$ $m=10$ $f=11$	$NR \rightarrow 8$ $R_0 \dots R_7 \text{--} R_f$ $f=8$	$NR \rightarrow 14$ $R_0 \dots R_9 \text{--} 4 \times R_m \text{--} R_f$ $m=\{10, 11, \dots 13\}$ $f=14$	$NR \rightarrow 5$ $R_0 \dots R_4 \text{--} R_f$ $f=5$	$NR \rightarrow 26$ $R_0 \dots R_9 \text{--} 16 \times R_m \text{--} R_f$ $m=\{10, 11, \dots 25\}$ $f=26$	$NR \rightarrow 42$ $R_0 \dots R_9 \text{--} 32 \times R_m \text{--} R_f$ $m=\{10, 11, \dots 41\}$ $f=42$	$NR \rightarrow 74$ $R_0 \dots R_9 \text{--} 64 \times R_m \text{--} R_f$ $m=\{10, 11, \dots 73\}$ $f=74$	$NR \rightarrow 138$ $R_0 \dots R_9 \text{--} 128 \times R_m \text{--} R_f$ $m=\{10, 11, \dots 137\}$ $f=138$				
	LOOP ARK COMEBACK	$NR \rightarrow 11$ $R_0 \dots R_9 \text{--} R_m \text{--} R_f$ $m=10$ $f=11$	$NR \rightarrow 8$ $R_0 \dots R_7 \text{--} R_f$ $f=8$	$NR \rightarrow 14$ $R_0 \dots R_9 \text{--} 4 \times R_m \text{--} R_f$ $m=\{10, 11, \dots 13\}$ $f=14$	$NR \rightarrow 6$ $R_0 \dots R_5 \text{--} R_f$ $f=6$	$NR \rightarrow 26$ $R_0 \dots R_9 \text{--} 16 \times R_m \text{--} R_f$ $m=\{10, 11, \dots 25\}$ $f=26$	$NR \rightarrow 42$ $R_0 \dots R_9 \text{--} 32 \times R_m \text{--} R_f$ $m=\{10, 11, \dots 41\}$ $f=42$	$NR \rightarrow 74$ $R_0 \dots R_9 \text{--} 64 \times R_m \text{--} R_f$ $m=\{10, 11, \dots 73\}$ $f=74$	$NR \rightarrow 138$ $R_0 \dots R_9 \text{--} 128 \times R_m \text{--} R_f$ $m=\{10, 11, \dots 137\}$ $f=138$				
	FOR	$NR \rightarrow 11$ $R_0 \dots R_9 \text{--} R_m \text{--} R_f$ $m=10$ $f=11$	$NR \rightarrow 8$ $R_0 \dots R_7 \text{--} R_f$ $f=8$	$NR \rightarrow 14$ $R_0 \dots R_9 \text{--} 4 \times R_m \text{--} R_f$ $m=\{10, 11, \dots 13\}$ $f=14$	$NR \rightarrow 6$ $R_0 \dots R_5 \text{--} R_f$ $f=6$	$NR \rightarrow 26$ $R_0 \dots R_9 \text{--} 16 \times R_m \text{--} R_f$ $m=\{10, 11, \dots 25\}$ $f=26$	$NR \rightarrow 42$ $R_0 \dots R_9 \text{--} 32 \times R_m \text{--} R_f$ $m=\{10, 11, \dots 41\}$ $f=42$	$NR \rightarrow 74$ $R_0 \dots R_9 \text{--} 64 \times R_m \text{--} R_f$ $m=\{10, 11, \dots 73\}$ $f=74$	$NR \rightarrow 138$ $R_0 \dots R_9 \text{--} 128 \times R_m \text{--} R_f$ $m=\{10, 11, \dots 137\}$ $f=138$				
	LOOP ARK COMEBACK	$NR \rightarrow 11$ $R_0 \dots R_9 \text{--} R_m \text{--} R_f$ $m=10$ $f=11$	$NR \rightarrow 8$ $R_0 \dots R_7 \text{--} R_f$ $f=8$	$NR \rightarrow 14$ $R_0 \dots R_9 \text{--} 4 \times R_m \text{--} R_f$ $m=\{10, 11, \dots 13\}$ $f=14$	$NR \rightarrow 7$ $R_0 \dots R_6 \text{--} R_f$ $f=7$	$NR \rightarrow 26$ $R_0 \dots R_9 \text{--} 16 \times R_m \text{--} R_f$ $m=\{10, 11, \dots 25\}$ $f=26$	$NR \rightarrow 42$ $R_0 \dots R_9 \text{--} 32 \times R_m \text{--} R_f$ $m=\{10, 11, \dots 41\}$ $f=42$	$NR \rightarrow 74$ $R_0 \dots R_9 \text{--} 64 \times R_m \text{--} R_f$ $m=\{10, 11, \dots 73\}$ $f=74$	$NR \rightarrow 138$ $R_0 \dots R_9 \text{--} 128 \times R_m \text{--} R_f$ $m=\{10, 11, \dots 137\}$ $f=138$				
	Table continued on next page												

Table continued on next page

Table A.2 – Table continued from previous page

Round counter	Attack moment	Model of injected fault									
		0x01	0x02	0x04	0x08	0x10	0x20	0x40	0x80		
		$R_{max} \rightarrow 11$		$R_{max} \rightarrow 8$	$R_{max} \rightarrow 14$		$R_{max} \rightarrow 2$	$R_{max} \rightarrow 26$	$R_{max} \rightarrow 42$	$R_{max} \rightarrow 74$	$R_{max} \rightarrow 138$
$RC = 7$											
	FOR	$N R \rightarrow 11$ $R_0 \dots R_9 \text{--} R_m \text{--} R_f$ $m=10$ $f=11$	$N R \rightarrow 8$ $R_0 \dots R_7 \text{--} R_f$ $f=8$	$N R \rightarrow 14$ $R_0 \dots R_9 \text{--} 4 \times R_m \text{--} R_f$ $m=\{10, 11, \dots 13\}$ $f=14$	$N R \rightarrow 7$ $R_0 \dots R_6 \text{--} R_f$ $f=7$	$N R \rightarrow 26$ $R_0 \dots R_9 \text{--} 16 \times R_m \text{--} R_f$ $m=\{10, 11, \dots 25\}$ $f=26$	$N R \rightarrow 42$ $R_0 \dots R_9 \text{--} 32 \times R_m \text{--} R_f$ $m=\{10, 11, \dots 41\}$ $f=42$	$N R \rightarrow 74$ $R_0 \dots R_9 \text{--} 64 \times R_m \text{--} R_f$ $m=\{10, 11, \dots 73\}$ $f=74$	$N R \rightarrow 138$ $R_0 \dots R_9 \text{--} 128 \times R_m \text{--} R_f$ $m=\{10, 11, \dots 137\}$ $f=138$		
	LOOP ARK COMEBACK	$N R \rightarrow 11$ $R_0 \dots R_9 \text{--} R_m \text{--} R_f$ $m=10$ $f=11$	$N R \rightarrow 8$ $R_0 \dots R_7 \text{--} R_f$ $f=8$	$N R \rightarrow 14$ $R_0 \dots R_9 \text{--} 4 \times R_m \text{--} R_f$ $m=\{10, 11, \dots 13\}$ $f=14$	$N R \rightarrow 8$ $R_0 \dots R_7 \text{--} R_f$ $f=8$	$N R \rightarrow 26$ $R_0 \dots R_9 \text{--} 16 \times R_m \text{--} R_f$ $m=\{10, 11, \dots 25\}$ $f=26$	$N R \rightarrow 42$ $R_0 \dots R_9 \text{--} 32 \times R_m \text{--} R_f$ $m=\{10, 11, \dots 41\}$ $f=42$	$N R \rightarrow 74$ $R_0 \dots R_9 \text{--} 64 \times R_m \text{--} R_f$ $m=\{10, 11, \dots 73\}$ $f=74$	$N R \rightarrow 138$ $R_0 \dots R_9 \text{--} 128 \times R_m \text{--} R_f$ $m=\{10, 11, \dots 137\}$ $f=138$		
$RC = 8$											
	FOR	$N R \rightarrow 11$ $R_0 \dots R_9 \text{--} R_m \text{--} R_f$ $m=10$ $f=11$	$N R \rightarrow 8$ $R_0 \dots R_7 \text{--} R_f$ $f=8$	$N R \rightarrow 14$ $R_0 \dots R_9 \text{--} 4 \times R_m \text{--} R_f$ $m=\{10, 11, \dots 13\}$ $f=14$	$N R \rightarrow 8$ $R_0 \dots R_7 \text{--} R_f$ $f=8$	$N R \rightarrow 26$ $R_0 \dots R_9 \text{--} 16 \times R_m \text{--} R_f$ $m=\{10, 11, \dots 25\}$ $f=26$	$N R \rightarrow 42$ $R_0 \dots R_9 \text{--} 32 \times R_m \text{--} R_f$ $m=\{10, 11, \dots 41\}$ $f=42$	$N R \rightarrow 74$ $R_0 \dots R_9 \text{--} 64 \times R_m \text{--} R_f$ $m=\{10, 11, \dots 73\}$ $f=74$	$N R \rightarrow 138$ $R_0 \dots R_9 \text{--} 128 \times R_m \text{--} R_f$ $m=\{10, 11, \dots 137\}$ $f=138$		
	LOOP ARK COMEBACK	$N R \rightarrow 11$ $R_0 \dots R_9 \text{--} R_m \text{--} R_f$ $m=10$ $f=11$	$N R \rightarrow 9$ $R_0 \dots R_8 \text{--} R_f$ $f=9$	$N R \rightarrow 14$ $R_0 \dots R_9 \text{--} 4 \times R_m \text{--} R_f$ $m=\{10, 11, \dots 13\}$ $f=14$	$N R \rightarrow 9$ $R_0 \dots R_8 \text{--} R_f$ $f=9$	$N R \rightarrow 26$ $R_0 \dots R_9 \text{--} 16 \times R_m \text{--} R_f$ $m=\{10, 11, \dots 25\}$ $f=26$	$N R \rightarrow 42$ $R_0 \dots R_9 \text{--} 32 \times R_m \text{--} R_f$ $m=\{10, 11, \dots 41\}$ $f=42$	$N R \rightarrow 74$ $R_0 \dots R_9 \text{--} 64 \times R_m \text{--} R_f$ $m=\{10, 11, \dots 73\}$ $f=74$	$N R \rightarrow 138$ $R_0 \dots R_9 \text{--} 128 \times R_m \text{--} R_f$ $m=\{10, 11, \dots 137\}$ $f=138$		

Table continued on next page

Table continued on next page

Table A.2 – Table continued from previous page

[illegible]

Bibliography

- [Agoyan 2010a] Michel Agoyan, Jean-Max Dutertre, Amir-Pasha Mirbaha, David Naccache, Anne-Lise Ribotta and Assia Tria. *Single-bit DFA using multiple-byte laser fault injection*. In International Conference on Technologies for Homeland Security – Proceedings of HST 2010, pages 113–119. IEEE, 2010. [166](#)
- [Agoyan 2010b] Michel Agoyan, Jean-Max Dutertre, David Naccache, Bruno Robisson and Assia Tria. *When clocks fail: On critical paths and clock faults*. In Smart Card Research and Advanced Application – Proceedings of CARDIS 2010, volume 6035 of LNCS, pages 182–193. Springer-Verlag, 2010. [12](#)
- [Amiel 2006] Frederic Amiel, Christophe Clavier and Michael Tunstall. *Fault Analysis of DPA-Resistant Algorithms*. In Fault Diagnosis and Tolerance in Cryptography – Proceedings of FDTC 2006, volume 4236 of LNCS, pages 223–236. Springer-Verlag, 2006. [12](#)
- [Anderson 1996] Ross Anderson and Markus Kuhn. *Tamper resistance - A cautionary note*. In USENIX Workshop on Electronic Commerce – Proceedings of EC 1996, pages 1–11. USENIX Association, 1996. [15](#)
- [Bar-El 2006] Hagai Bar-El, Hamid Choukri, David Naccache, Michael Tunstall and Claire Whelan. *The Sorcerer’s Apprentice Guide to Fault Attacks*. Proceedings of IEEE, vol. 94, no. 2, pages 370–382, 2006. [7](#), [10](#), [14](#), [19](#), [21](#), [153](#)
- [Barengi 2009] Alessandro Barengi, Guido Bertoni, Emanuele Parrinello and Gerardo Pelosi. *Low Voltage Fault Attacks on the RSA Cryptosystem*. In Fault Diagnosis and Tolerance in Cryptography – Proceedings of FDTC 2009, pages 23–31. IEEE, 2009. [11](#)
- [Barengi 2010] Alessandro Barengi, Guido M. Bertoni, Luca Breveglieri, Mauro Pellicoli and Gerardo Pelosi. *Low Voltage Fault Attacks to AES*. In Hardware-Oriented Security and Trust – Proceedings of HOST 2010, pages 7–12. IEEE, 2010. [10](#), [153](#)
- [Barengi 2012] Alessandro Barengi, Luca Breveglieri, Israel Koren and David Naccache. *Fault Injection Attacks on Cryptographic Devices: Theory, Practice and Countermeasures*. Proceedings of the IEEE, vol. 100, no. 11, pages 3056–3076, 2012. [7](#), [11](#), [12](#), [13](#), [14](#), [15](#)
- [Bellcore 1996] Bellcore. *New threat model breaks crypto codes*. 1996. [5](#), [8](#), [149](#), [152](#)
- [Biham 1997] Eli Biham and Adi Shamir. *Differential Fault Analysis of Secret Key Cryptosystems*. In Advances in Cryptology – Proceedings of CRYPTO 1997, volume 1294 of LNCS, pages 513–525. Springer-Verlag, 1997. [5](#), [8](#), [149](#), [152](#)

- [Blömer 2003] Johannes Blömer and Jean-Pierre Seifert. *Fault Based Cryptanalysis of the Advanced Encryption Standard (AES)*. In Financial Cryptography – Proceedings of FC 2003, volume 2742 of LNCS, pages 162–181. Springer-Verlag, 2003. 9, 20, 55, 152
- [Bogdanov 2011] Andrey Bogdanov, Dmitry Khovratovich and Christian Rechberger. *Biclique Cryptanalysis of the Full AES*. Cryptology ePrint Archive, Report 2011/449, 2011. <http://eprint.iacr.org/>. 4
- [Boneh 1997] Dan Boneh, Richard DeMillo and Richard Lipton. *On The Importance of Checking Cryptographic Protocols For Faults*. In Theory and Application of Cryptographic Techniques – Proceedings of EuroCrypt 1997, volume 1233 of LNCS, pages 37–51. Springer-Verlag, 1997. 5, 8, 149, 152
- [Bruestle 2002] Matthias Bruestle. *SOSSE - Simple Operating System for Smartcard Education*. <http://www.mbsks.franken.de/sosse/index.html>, 2002. 27, 28, 159, 160
- [Chen 2003] Chien-Ning Chen and Sung-Ming Yen. *Differential Fault Analysis on AES Key Schedule and Some Countermeasures*. In Information Security and Privacy – Proceedings of ACISP 2003, volume 2727 of LNCS, pages 118–129. Springer-Verlag, 2003. 20
- [Choukri 2005] Hamid Choukri and Michael Tunstall. *Round Reduction Using Faults*. In Fault Diagnosis and Tolerance in Cryptography – Proceedings of FDTCT 2005, pages 13–24, 2005. 8, 20, 77, 78, 151, 152, 167, 168
- [CNSS 2003] CNSS. *National policy on the use of the advanced encryption standard (AES) to protect national security systems and national security information*. volume 15 of CNSS Policy. Committee on National Security Systems (CNSS), 2003. 17
- [CNSS 2010] CNSS. *National Information Assurance (IA) Glossary*. volume 4009 of CNSS Instruction. Committee on National Security Systems (CNSS), 2010. 119
- [Darracq 2002] F. Darracq, T. Beauchene, V. Pouget, H. Lapuyade, D. Lewis, P. Fouillat and A. Touboul. *Single-event sensitivity of a single SRAM cell*. IEEE Transactions on Nuclear Science, vol. 49, no. 3, pages 1486–1490, 2002. 7, 19, 21, 157
- [Denning 1983] Dorothy Elisabeth Denning. *Cryptography and data security*. Addison-Wesley, 1983. 2, 148
- [Dusart 2003] Pierre Dusart, Gilles Letourneux and Olivier Vivolo. *Differential Fault Analysis on A.E.S.* In Applied Cryptography and Network Security – Proceedings of ACNS 2003, volume 2846 of LNCS, pages 293–306. Springer-Verlag, 2003. 20, 55

- [Dutertre 2010] Jean-Max Dutertre, Amir-Pasha Mirbaha, Assia Tria, Bruno Robisson and Michel Agoyan. *Revue expérimentale des techniques d'injection de fautes*. In GDR SoC-SiP - Journée sécurité. http://www.lirmm.fr/journees_securite/material/j2/Dutertre.pdf, 2010. 12
- [Fibics] Fibics. *Introduction: Focused Ion Beam Systems*. Fibics Incorporated. <http://www.fibics.com/fib/tutorials/introduction-focused-ion-beam-systems/4/index.html>. 15
- [Gandolfi 2001] Karine Gandolfi, Christophe Mourtel and Francis Olivier. *Electromagnetic Analysis: Concrete Results*. In Cryptographic Hardware and Embedded Systems – Proceedings of CHES 2001, volume 2162 of *Lecture Notes in Computer Science*, pages 251–261. Springer, 2001. 5, 149
- [Giraud 2005] Christophe Giraud. *DFA on AES*. In Advanced Encryption Standard – Proceedings of AES 2004, volume 3373 of LNCS, pages 27–41. Springer-Verlag, 2005. 13, 20, 42, 49, 51, 52, 55, 122, 164
- [Govindavajhala 2003] Sudhakar Govindavajhala and Andrew W. Appel. *Using Memory Errors to Attack a Virtual Machine*. In Security and Privacy – Proceedings of SP 2003, pages 154–165. IEEE, 2003. 12, 14
- [ISO/IEC 7816-3 2003] ISO/IEC 7816-3. ISO/IEC 7816-3:2003 information technology - identification cards - integrated circuit(s) cards with contacts - part 3: Electronic signals and transmission protocols. International Organization for Standardization (ISO) / International Electrotechnical Commission (IEC), 2003. 6, 28, 137, 139, 150
- [Joye 2005] M. Joye and F. Olivier. *Side Channel Analysis*. In Encyclopedia of Cryptography and Security, pages 571–576. Kluwer Academic Publishers, 2005. 5, 149
- [Kahn 1973] David Kahn. *The codebreakers - the story of secret writing*. Macmillan, 1973. 3, 149
- [Kim 2007] Chong Hee Kim, Jong Hoon Shin, Jean-Jacques Quisquater and Pil Joong Lee. *Safe-Error Attack on SPA-FA Resistant Exponentiations Using a HW Modular Multiplier*. vol. 4817, pages 273–281, 2007. 9
- [Kim 2008] Chong Kim and Jean-Jacques Quisquater. *New Differential Fault Analysis on AES Key Schedule: Two Faults Are Enough*. In Smart Card Research and Advanced Applications, volume 5189 of LNCS, pages 48–60. Springer, 2008. 20
- [Kocher 1996] Paul C. Kocher. *Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems*. In Advances in Cryptology – Proceedings of CRYPTO 1996, volume 1109 of LNCS, pages 104–113. Springer-Verlag, 1996. 4, 134, 149

- [Kocher 1998] Paul Kocher. *Introduction to Differential Power Analysis and Related Attacks*. <http://www.cryptography.com/public/pdf/DPATechInfo.pdf>, 1998. 134
- [Kocher 1999] Paul C. Kocher, Joshua Jaffe and Benjamin Jun. *Differential Power Analysis*. In *Advances in Cryptology – Proceedings of CRYPTO 1999*, volume 1666 of LNCS, pages 388–397. Springer-Verlag, 1999. 5, 134
- [Kömmerling 1999] Oliver Kömmerling and Markus G. Kuhn. *Design Principles for Tamper-Resistant Smartcard Processors*. In *USENIX Workshop on Smartcard Technology – Proceedings of Smartcard 1999*, pages 9–20. USENIX Association, 1999. 15
- [Konheim 2007] Alan G. Konheim. *Computer security and cryptography*. John Wiley & Son, 2007. 2, 149
- [Li 2010] Yang Li, Kazuo Sakiyama, Shigeto Gomisawa, Toshinori Fukunaga, Junko Takahashi and Kazuo Ohta. *Fault Sensitivity Analysis*. In *Cryptographic Hardware and Embedded Systems – Proceedings of CHES 2010*, volume 6225 of LNCS, pages 320–334. Springer-Verlag, 2010. 10, 153
- [Lu 2005] Chih-Chung Lu, Shau-Yin Tseng and Szu-Kai Huang. *A secure modular exponential algorithm resists to power, timing, C safe error and M safe error attacks*. In *Advanced Information Networking and Applications – Proceedings of AINA 2005*, pages 151–154. IEEE, 2005. 9
- [Mayes 2008] Keith E. Mayes. *An Introduction to Smart Cards*. In *Smart Cards, Tokens, Security and Applications*, pages 1–25. Springer, 2008. 28
- [Melinger 1994] J.S. Melinger, S. Buchner, D. McMorrow, W.J. Stapor, T.R. Weatherford and A.B. Campbell. *Critical evaluation of the pulsed laser method for single event effects testing and fundamental studies*. vol. 41, no. 6, pages 2574–2584, 1994. 22
- [Melngailis 1986] J. Melngailis, C. R. Musil, E. H. Stevens, M. Utlaut, E. M. Kellogg, R. T. Post, M. W. Geis and R. W. Mountain. *The focused ion beam as an integrated circuit restructuring tool*. *Journal of Vacuum Science & Technology B: Microelectronics and Nanometer Structures*, vol. 4, no. 1, pages 176–180, 1986. 15
- [Mollin 2007] Richard A. Mollin. *An introduction to cryptography - second edition*. Chapman & Hall, 2007. 1, 148
- [Monnet 2006] Yannick Monnet, Marc Renaudin, Régis Leveugle, Christophe Clavier and Pascal Moitrel. *Case Study of a Fault Attack on Asynchronous DES Crypto-Processors*. In *Fault Diagnosis and Tolerance in Cryptography – Proceedings of FDTC 2006*, volume 4236 of LNCS, pages 88–97. Springer-Verlag, 2006. 8, 20, 77, 78, 168

- [Moradi 2006] Amir Moradi, Mohammad Shalmani and Mahmoud Salmasizadeh. *A Generalized Method of Differential Fault Attack Against AES Cryptosystem*. In Cryptographic Hardware and Embedded Systems – Proceedings of CHES 2006, volume 4249 of LNCS, pages 91–100. Springer-Verlag, 2006. 49
- [Nisene] Nisene. *JetEtch*. Nisene Technology Group. <http://www.nisene.com/jetetch.shtml> and <http://www.nisene.com/jetetch2.shtml>. 32
- [NIST 2001] NIST. Announcing the advanced encryption standard (AES) - FIPS 197. National Institute of Standards and Technology (NIST), 2001. 17, 78, 155, 156, 168
- [Otto 2004] Martin Otto. *Fault Attacks and Countermeasures*. PhD thesis, 2004. 14, 16, 154
- [Paar 2010] Christof Paar and Jan Pelzl. Understanding cryptography. Springer, 2010. 1, 3, 148, 149
- [Park 2011] JeaHoon Park, SangJae Moon, DooHo Choi, YouSung Kung and JaeCheol Ha. *Differential fault analysis for round-reduced AES by fault injection*. ETRI Journal, vol. 33, no. 3, pages 434–442, 2011. 8, 20, 77, 78, 168
- [Pavlov 2008] A. Pavlov and M. Sachdev. CMOS SRAM circuit design and parametric test in nano-scaled technologies, volume 40 of *Frontiers in Electronic Testing*. Springer-Verlag, 2008. 60
- [Piret 2003] Gilles Piret and Jean-Jacques Quisquater. *A Differential Fault Attack Technique against SPN Structures, with Application to the AES and KHAZAD*. In Cryptographic Hardware and Embedded Systems – Proceedings of CHES 2003, volume 2779 of LNCS, pages 77–88. Springer-Verlag, 2003. 20, 42, 43, 44, 55, 164
- [Rankl 2003] Wolfgang Rankl and Wolfgang Effing. Smart card handbook - 3rd edition. John Wiley & Sons, 2003. 28
- [Richter 1987] A.K. Richter and I. Arimura. *Simulation of Heavy Charged Particle Tracks Using Focused Laser Beams*. IEEE Transactions on Nuclear Science, vol. 34, no. 2, pages 1234–1239, 1987. 7
- [Robisson 2007] Bruno Robisson and Pascal Manet. *Differential Behavioral Analysis*. In Cryptographic Hardware and Embedded Systems – Proceedings of CHES 2007, volume 4727 of LNCS, pages 413–426. Springer-Verlag, 2007. 9, 20, 152, 153
- [Sanyo 2008] Sanyo. *Sanyo Amorphous Solar Cell - Amorton - AM-8801*. http://semicon.sanyo.com/en/ds_e/ENA0562A.pdf, 2008. 137, 178

- [Schmidt 2007] J.M. Schmidt and M. Hutter. *Optical and EM fault-attacks on CRT-based RSA: Concrete results*. In Austrian Workshop on Microelectronics – Proceedings of Austrochip 2007, pages 61–67. Verlag der Technischen Universität, 2007. 12, 13
- [Schmidt 2009] J.M. Schmidt, M. Hutter and T. Plos. *Optical and EM fault-attacks on AES: A threat in violet*. In Fault Diagnosis and Tolerance in Cryptography – Proceedings of FDTC 2009, pages 13–22. IEEE, 2009. 13
- [Schwank 2006] J.R. Schwank, M.R. Shaneyfelt, J.A. Felix, P.E. Dodd, J. Baggio, V. Ferlet-Cavrois, P. Paillet, G.L. Hash, R.S. Flores, L.W. Massengill and E. Blackmore. *Effects of Total Dose Irradiation on Single-Event Upset Hardness*. IEEE Transactions on Nuclear Science, vol. 53, no. 4, pages 1772–1778, 2006. 14
- [Selmane 2008] Nidhal Selmane, Sylvain Guilley and Jean-Luc Danger. *Practical Setup Time Violation Attacks on AES*. In European Dependable Computing Conference – Proceedings of EDCC 2008, pages 91–96. IEEE, 2008. 11
- [Semiconductors 2009] Osram Opto Semiconductors. *Orbeos for OLED lighting / Preliminary Data*. Nisene Technology Group. http://www.osram-os.com/osram_os/EN/Products/Product_Promotions/OLED_Lighting/_pdf/Datasheet_ORBEOS_CDW-031.pdf, 2009. 135, 178
- [Skorobogatov 2003] Sergei Skorobogatov and Ross Anderson. *Optical Fault Induction Attacks*. In Cryptographic Hardware and Embedded Systems – Proceedings of CHES 2002, volume 2523 of LNCS, pages 31–48. Springer-Verlag, 2003. 13, 19, 37, 156
- [Skorobogatov 2005] Sergei Skorobogatov. *Semi-invasive attacks - A new approach to hardware security analysis*. Technical report UCAM-CL-TR-630, Computer Laboratory, University of Cambridge, 2005. 13
- [Standaert 2009] François-Xavier Standaert. *Secure and Efficient Implementation of Symmetric Encryption Schemes using FPGAs*. In Çetin Kaya Koç, editeur, Cryptographic Engineering, pages 295–320. Springer, 2009. 6
- [Torrance 2009] Randy Torrance and Dick James. *The State-of-the-Art in IC Reverse Engineering*. In Cryptographic Hardware and Embedded Systems – Proceedings of CHES 2009, volume 5747 of LNCS, pages 363–381. Springer-Verlag, 2009. 15
- [Tria 2000] Assia Tria, Bruno Robisson, Jean-Max Dutertre and Amir-Pasha Mirbaha. *Fault attacks from theory to practise: what is possible to do?* In 2-nd Canada-France Workshop on Foundations & Practice of Security. http://www-mitacs2009.imag.fr/Material/mitac_part1.pdf and http://www-mitacs2009.imag.fr/Material/mitac_part2.pdf, 2000. 5, 150

- [Tunstall 2011] Michael Tunstall, Debdeep Mukhopadhyay and Subidh Ali. *Differential Fault Analysis of the Advanced Encryption Standard Using a Single Fault*. In Information Security Theory and Practice. Security and Privacy of Mobile Devices in Wireless Communication, volume 6633 of LNCS, pages 224–233. Springer, 2011. 20
- [Ultra Tec] Ultra Tec. *ASAP-1 Decap*. Ultra Tec Manufacturing Inc. <http://www.ultratecusa.com/asap-1-decap>. 32
- [Vacca 2009] John R. Vacca. Computer and information security handbook. Morgan Kaufmann, 2009. 119
- [Van Tilborg 2005] Henk C. A. Van Tilborg. Encyclopedia of cryptography and security. Springer Science+Business Media, 2005. 2, 4, 5, 148, 149
- [Yen 2000] Sung-Ming Yen and Marc Joye. *Checking Before Output May Not Be Enough Against Fault-Based Cryptanalysis*. IEEE Transactions on Computers, vol. 49, no. 9, pages 967–970, 2000. 9, 152
- [Yen 2002] Sung-Ming Yen, Seungjoo Kim, Seongan Lim and Sangjae Moon. *A countermeasure against one physical cryptanalysis may benefit another attack*. vol. 2288, pages 269–294, 2002. 9

École Nationale Supérieure des Mines
de Saint-Étienne

NNT: 2011 EMSE 0636

Amir-Pasha MIRBAHA

**STUDY OF THE VULNERABILITY OF CRYPTOGRAPHIC
CIRCUITS BY LASER FAULT INJECTION**

Specialty: Microelectronics

Keywords: Fault Attack, Physical Attack, Laser Fault Injection, Cryptography, Smart Card, Advanced Encryption Standard, Differential Fault Analysis, Round Modification Analysis.

Abstract: Cryptographic circuits may be victims of fault attacks on their hardware implementations. Fault attacks consist of creating intentional faults during cryptographic calculations in order to reveal secrets. In the context of the security characterization of circuits, we examined the practical feasibility of some theoretical models of fault attacks. We used a laser bench as a means of fault injection.

In the beginning, we performed laser fault injections on a microcontroller implementing an AES cryptographic algorithm. We succeeded in excluding the logical effect of mismatched faults by temporal and spatial accuracy in fault injection. Moreover, we identified extended new DFA attacks.

Then, we extended our research to identify and to implement new fault attack models. With the precision obtained in our earlier work, we developed new Round Modification Analysis (RMA) attacks.

In conclusion, previous works have warned on the feasibility of the fault attacks, described in the literature, by laser fault injection. Our tests have demonstrated that single-byte or single-bit attacks are still feasible with a laser beam which hits several bytes. Our experiments also revealed new attack possibilities with their analysis methods. These results led us to the study of appropriate countermeasures.

École Nationale Supérieure des Mines
de Saint-Étienne

NNT : 2011 EMSE 0636

Amir-Pasha MIRBAHA

**ÉTUDE DE LA VULNÉRABILITÉ DES CIRCUITS
CRYPTOGRAPHIQUES À L'INJECTION DE FAUTES PAR LASER**

Spécialité : Microélectronique

Mots clefs : Attaque en faute, Attaque matérielle, Injection de fautes par laser, Cryptographie, Carte à puce, Advanced Encryption Standard, Analyse différentielle des fautes, Analyse de modification de rondes.

Résumé : Les circuits cryptographiques peuvent être victimes d'attaques en fautes visant leur implémentation matérielle. Elles consistent à créer des fautes intentionnelles lors des calculs cryptographiques afin d'en déduire des informations confidentielles. Dans le contexte de la caractérisation sécuritaire des circuits, nous avons été amenés à nous interroger sur la faisabilité expérimentale de certains modèles théoriques d'attaques. Nous avons utilisé un banc laser comme moyen d'injection de fautes.

Dans un premier temps, nous avons effectué des attaques en fautes DFA par laser sur un microcontrôleur implémentant un algorithme de cryptographie AES. Nous avons réussi à exclure l'effet logique des fautes ne correspondant pas aux modèles d'attaque, par un jeu précis sur l'instant et le lieu d'injection. En outre, nous avons identifié de nouvelles attaques DFA plus élargies.

Ensuite, nous avons étendu nos recherches à la découverte et la mise en place de nouveaux modèles d'attaques en fautes. Grâce à la précision obtenue lors de nos premiers travaux, nous avons développé des nouvelles attaques par modification de rondes.

En conclusion, les travaux précédents constituent un avertissement sur la faisabilité avérée des attaques par laser décrites dans la littérature scientifique. Nos essais ont témoigné de la faisabilité toujours actuelle de la mise en place des attaques mono-octets ou mono-bits avec un faisceau de laser qui rencontre plusieurs octets ; et également, révéler de nouvelles possibilités d'attaque et de leurs méthodes d'analyse. Cela nous a amenés à étudier des contre-mesures adaptées.
