Etude de la vulnérabilité des circuits cryptographiques l'injection de fautes par laser.

Amir-Pasha Mirbaha

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 :	**Responsables :**
SCIENCES ET GENIE DES MATERIAUX	K. Wolski Directeur de recherche
MECANIQUE ET INGENIERIE	S. Drapier, professeur
GENIE DES PROCEDES	F. Gruy, Maître de recherche
SCIENCES DE LA TERRE	B. Guy, Directeur de recherche
SCIENCES ET GENIE DE L'ENVIRONNEMENT	D. Graillot, Directeur de recherche
MATHEMATIQUES APPLIQUEES	O. Roustant, Maître-assistant
INFORMATIQUE	O. Boissier, Professeur
IMAGE, VISION, SIGNAL	JC. Pinoli, Professeur
GENIE INDUSTRIEL	A. Dolgui, Professeur
MICROELECTRONIQUE	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
DELAFOSSE	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
GOEURIOT	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
KLÖCKER	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	SMS
XIE	Xiaolan	PR 1	Génie industriel	CIS

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 1ère classe	MCF	Maître de conférences
PR 2	Professeur 2ème 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

Dernière mise à jour le : 02 avril 2012

Ecole Nationale
Supérieure des Mines
SAINT-ETIENNE

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:

President:	Régis Leveugle	Professor, TIMA, Grenoble
Reviewers:	Philippe Maurine	Associate Professor, LIRMM, Montpellier
	Jean-Jacques Quisquater	Professor, UCL, Louvain-la-Neuve
Advisors:	Assia Tria	Head of Laboratory, CEA-LETI, Gardanne
	David Naccache	Professor, ENS, Paris
Co-Advisor:	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 son bon 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 Reymond, Patrick Orsatelli, Nicolas Moro, Hélène Le Bouder, 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, Professeur George Malliaras, Dion Khodagholy, Etienne Drahi, Gaëlle Rondeau, Pauline Sajous, Béatrice Dubois, Sylvain Nolot, Thierry Camilloni, Cyril Calmes et Sylvain Blayac que j'ai plusieurs fois consultés lors de mes travaux pour les agrémenter 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 suis 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 Professeur Christophe Donnet, Professeur 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 Béheshti qui m'a toujours encouragé et soutenu pour accomplir mes études de doctorat et pour m'avoir orienté vers le 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 Paqueteau, l'Attaché scientifique et technique et son successeur, Dr. Sixte Blanchy ; ainsi que Zohreh Mirbaha, la Responsable du Service des étudiants et mon ancien professeur, Armand Karimi Goudarzi 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 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 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

List of Figures

List of Tables

CHAPTER 1

Introduction

Contents

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 that can remove 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 family, messages are encrypted and decrypted using a unique secret key that provides the 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 using the given key to a corresponding ciphertext:

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

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

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

In asymmetric cryptography, the key used for decryption is different from the encryption key. Besides, the 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 shortly *cryptosystems*. They were originally performed by hand methods using pen-and-pencil calculations. In the 18th, mechanical machine methods were introduced to speed up the 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 are 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 to 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 revealed. In the symmetric methods, the same encryption key is used for the decryption. Therefore, the encryption key must be protected. Besides, in the asymmetric cryptography, the encryption key can be exposed.

So, the security of the cipher is based on the security of the decryption key in the 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 to deduct the keys by mathematical methods. When an opponent is not able to find any weakness in a cryptosystem that would help him to 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 needs checking in average the of the entire key search space.

   The key length of reliable cryptographic algorithms increases continually above the progresses of calculation capability of computers for finding the keys. So, a brute-force search for their 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 great 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 in that period. But, accordingly to the growth of computation capabilities, the US National Institute of Standards and Technology (NIST) announced in 1997 that they wish to choose a successor to DES. They mentioned that the new standard will be known as *Advanced Encryption Standard* or AES. They also remark 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 three years of studies and discussions.

The growth of minimal key size of the US encryption standard from 56 bits to 128 bits during 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 exhaustive search.

Another significant example is the success of a new research reported in [Bogdanov 2011]. The method decreases the complexity of exhaustive search for AES key variants for only two bits. In 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 needs still thousands years with 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, related to sensitive data processing that can reveal the secret key. For instance, the running time of a cryptographic circuit may leak informations 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 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] have 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 fail 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, 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 for performing 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 informations that permits side-channel analysis.

This kind of integrated circuits are considered as *non-secure* for cryptographic systems and they are highly not recommended for these purposes.

Besides, there are specific integrated circuits for cryptographic systems that resist more at physical extreme conditions and 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 to deduct any sensitive information. These circuits are called *secure systems*. This kind of protections may applied to side-channel leakages.

For example, a bank card with integrated chip is a kind of secure systems. According to ISO/IEC 7816-3 standards the chip operations interrupts 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 a data masking protection by random calculations on their chip. 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 prevent an attack to succeed. 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 informations.

According to [Standaert 2009], any single technique cannot allow to provide perfect security. Protecting implementations against physical attacks consequently intends 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, we will present some countermeasures in the chapter 5.

1.1.3 Different Types of Faults

Faults on the electronic circuits can be categorized according to their persistence under three classes:

- Provisional or transient faults: These faults are temporary or short-term. As the fault injection interrupts, the provisional faults disappear. So, after elapsing a several amount of time, 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. So, 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. But, the memory must be rewritten.

- Destructive faults: The interferences may create a perpetual defect on hardware. Once inflicted, such destructions affects the chip's behavior permanently. For example, a laser emission with high energy level on a memory cell may destroy permanently some memory cells. In this case, the memory cells cannot be any more rewritten or recovered by circuit reset.

Different physical reasons are identified as origin of faults in each class [Bar-El 2006] [Barenghi 2011]. We summarize them in the subsequent section. Now, we describe briefly main permanent and destructive faults in secure circuits.

1.1.3.1 Permanent Faults

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

- Single-event upsets (SEUs): This kind of faults was first noticed as an effect of cosmic rays during a space mission in 1975. Then, researches are begun on injection mechanisms of such faults on 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 researches are 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 permanently and creates lifetime faulty value or behavior. Different types of faults are included in this class [Bar-El 2006]:

- Single-event snap back faults (SESs): This kind of faults is created due to the self-sustained current by the parasitic bipolar transistor in channel N of MOS transistors. It seems that they does 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 requires injection of transient or permanent faults. Otherwise, if the opponent creates destructive faults, the procedure can not 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 theses algorithms' strength against cryptanalysis is based on their repeated rounds. Any decrease on the number of rounds reduces their security.

  The Round Reduction belongs to the family of attacks by algorithm modification. For instance, suppose an attack by the opponent that makes a jump after execution of few instructions or the first round at the beginning of algorithm to its end. So, the remaining encryption processes are skipped and the final ciphertext is the product of few algorithm processes that may reveal easily 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 more corresponds to the cryptanalysis of correct execution of entire 10 rounds for 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 the chapter 4.

- **Differential Fault Analysis** (DFA): This method is based on gaining some insights 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 about 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 the chapter 3.

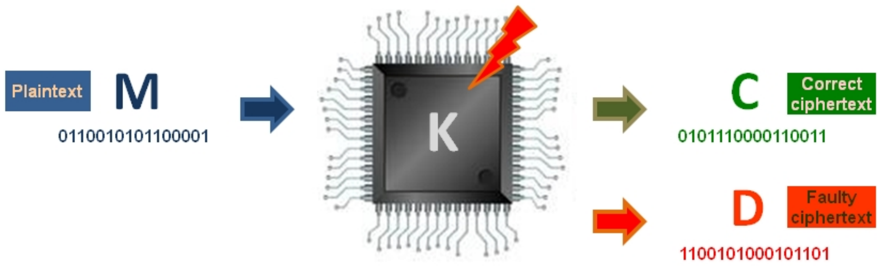


Figure 1.1: An overview of Differential Fault Analysis (DFA).

- **Safe-Error Analysis** (SEA): This analysis method searches for existence of any behavioral difference of a circuit instead of faulty chiphertexts. A fault attack, may release an alarm or stop the operations. These signs of a behavioral difference in comparison with a normal execution may lead to find secrets from 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 attacked 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] are distinguished between the two attacks, by considering the first method as a *Memory* or M *Safe-Error* that targets memory or register contents and the second one as a *Computational* or C *Safe-Error* Analysis focusing on the operations. Besides, [Kim 2007] reports vulnerability of a smart card implemented with a hardware modular multiplier to another type of SEA.

  In 2003, Bl omer and Seifert presented two SEA on 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 condition when a faulty output begins to exhibit some detectable characteristics.

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

- Voltage glitch and clock modification may applied to the corresponding pins on the chip.

- Temperature increase and electromagnetic impulsion might be applied without chip decapsulation.

- High-energy light emission, 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 of the implementation for the opponent.

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

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 include also group of experts or companies with a good budget that help to acquire a very good knowledge of a system.

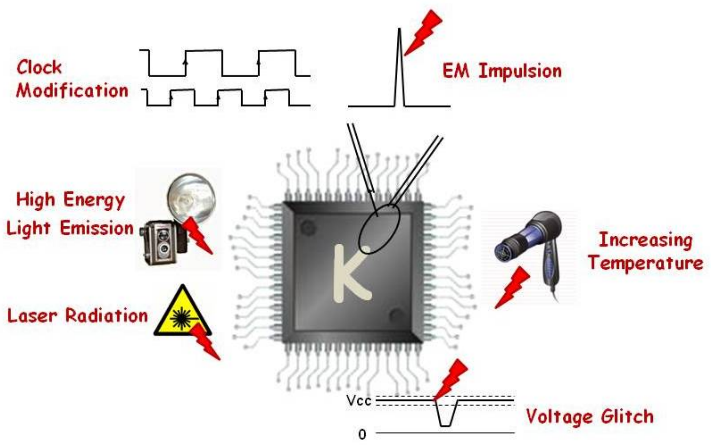


Figure 1.2: An overview of fault injection techniques.

1.1.5.1 Low cost fault injection methods

[Barenghi 2011] considers as *low cost* any fault injection technique that need to less than 3000 USD of equipment in order to set up the attack. This budget is estimated in the ability of a single motivated opponent. So, these fault injection methods are usually intended as a serious threat. Consequently, the implementations of recent secure chips have mostly proper countermeasures against them and they cannot alter their security.

1. Underpowering the circuit during execution may cause a processor to interrupt or skip instruction. Through 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 such as an ARM9 processor. As reported in [Barenghi 2009], underpowering of the circuit entered delays on the set-up time of its logical gates. Therefore, faults are injected on the circuit computations. Another successful attack is reported in [Selmane 2008] on a smart card embedding a symmetrical encryption algorithm.

   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, it requires only basic skills and can be easily achieved without evidence of tampering. Besides, any detailed knowledge of the device's implementation is not required.[Barenghi 2011]

2. Tampering the clock signal is another option to cause 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 can not be executed correctly.

   If, for instance, a data read is requested from the memory and it has not time to latch out the content on the bus, a data misread happens. Or, if before finishing the execution of a current instruction, the shorter clock signal cause jumping to the next instruction, it 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. But, the opponent needs to have direct control on the clock line. For the chips that generate their own clock signal internally, disconnecting the clock line and performing a such attack seems to be infeasible [Barenghi 2011].

3. Altering the environmental conditions is another possibility to inject faults. For instance, by increasing the temperature, it is possible to create multiple 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 about 10 flipped bits per 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 equipments.

   Heating a circuit lengthens propagation delays in CMOS logical circuits. If the propagation delays in a circuit becomes 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 at 210°C using an electrical heater of compressed air.

   For this kind of attacks, a minimal technical skill suffices. But, the possibility of damaging the circuit by excessive temperature can be considered as potential risk and disadvantage of temperature attacks.

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, settled very close to the targeted chip. [Schmidt 2007] reports also 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 was held directly above the device.

Besides, in this kind of attacks, 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 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 needed for an 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 depending on the duration of the irradiation process.

1.1.5.2 High cost fault injection methods

According to [Barenghi 2011], a second class of fault injection techniques exists for opponents with a bigger budget. These techniques are developed for the need to have a direct access to the silicon die and the ability to target individual circuits with accuracy. 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 light beam [Barenghi 2011].

   [Skorobogatov 2003] and [Skorobogatov 2005] report successful targeting of an SRAM cell 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 more possible to hit a single SRAM cell on the circuits with current etching technology.

2. Laser emission on a decapsulated chip is another way to generate faults. Using a laser beam allows to target more precisely a small circuit area. It can be considered as the most straightforward refinement of the previous technique [Barenghi 2011]. 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 big different accuracy and also prices from few thousands to several hundred thousands USD.

   This method has also some limitations: The incident area on a targeted chip can not reduced to a smaller size than the laser wavelength. Therefore, depending to the fabrication technology, the incident area is physically larger than one gate on 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 researches, such as [Schwank 2006] have confirmed the impacts of X-rays, as a cosmic ray, 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 impacts 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 equipments that are more expensive than low cost methods. However, as the X-rays have a very short wavelengths in the range of 0.01 to 10nm, it might be possible that they could inject accurate faults. The future researches will determine it.

4. Using Focused Ion Beams (FIB) permits an opponent to perform the most accurate and powerful fault injection attack at the present time [Barenghi 2011]. 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 engineer unknown 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 interconnects. That is not fault injection but rather *circuit spying*.

[Melngailis 1986] has proposed to explore the ion milling capability of FIBs as a tool for integrated circuit reconstructing in 1986. It has illustrated two methods for cutting and joining conductors on the circuits. Afterwards, other researches such as [Anderson 1996] and [Kömmerling 1999] confirmed 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 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 cost very expensive. They also require a strong technical background to fully exploit their capabilities. According to [Barenghi 2011], 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.

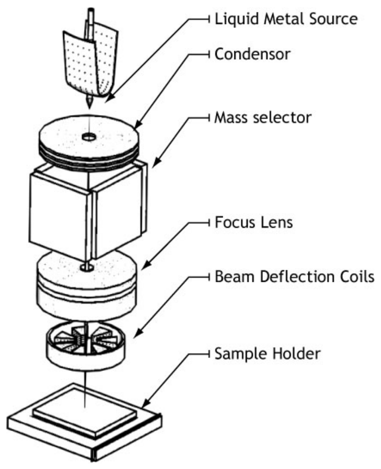


Figure 1.3: Scheme of a FIB system.

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 $T_1 = \{b_1, b_2, ...b_n\}$ as the initial values of an arbitrary set of targeted bits. Let $T_2 = \{b'_1, b'_2, ...b'_n\}$ be values of T_1 after fault attack. Now, we review 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:

$$\forall i : 0 \leq i \leq n \quad ; \quad b'_i = 1 - b_i \tag{1.1}$$

- **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 disappear after a circuit reset.

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

$$\forall i : 0 \leq i \leq n \quad ; \quad b_i = b'_i = 0 \tag{1.2}$$

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

$$\forall i : 0 \leq i \leq n \quad ; \quad b_i = b'_i = 1 \tag{1.3}$$

  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. It 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:

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

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

$$\forall i : 0 \leq i \leq n \quad ; \quad b'_i = 1 \tag{1.5}$$

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

$$\forall i : 0 \leq i \leq n \quad ; \quad b'_i = 0 \tag{1.6}$$

Between these fault models, the random faults are usually considered as 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]. It 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.

On June 2003, US National Security Agency (NSA) has 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 very substantial for AES systems security. Today, a significant part of researches in cryptography is focused on improving the cryptographic algorithm and systems against any eventual attack.

AES is an algorithm that performs message encryption processing by data blocks of 128 bits at input and output using a key size of 128, 192 or 256 bits respectively in 10, 12 or 14 rounds (after a short initial round) according to the size of the key. The algorithm includes two separated processes: One for the key scheduling to derive the round keys from the secret key and another one for the data encryption.

Decryption also is divided into two separated processes: One for the `KeyScheduling` and another one for the `DataDecryption`.

For the initial round in AES-128, 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 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 9-th round). For AES versions, we will mention their key lengths (i.e. AES-192 and AES-256).

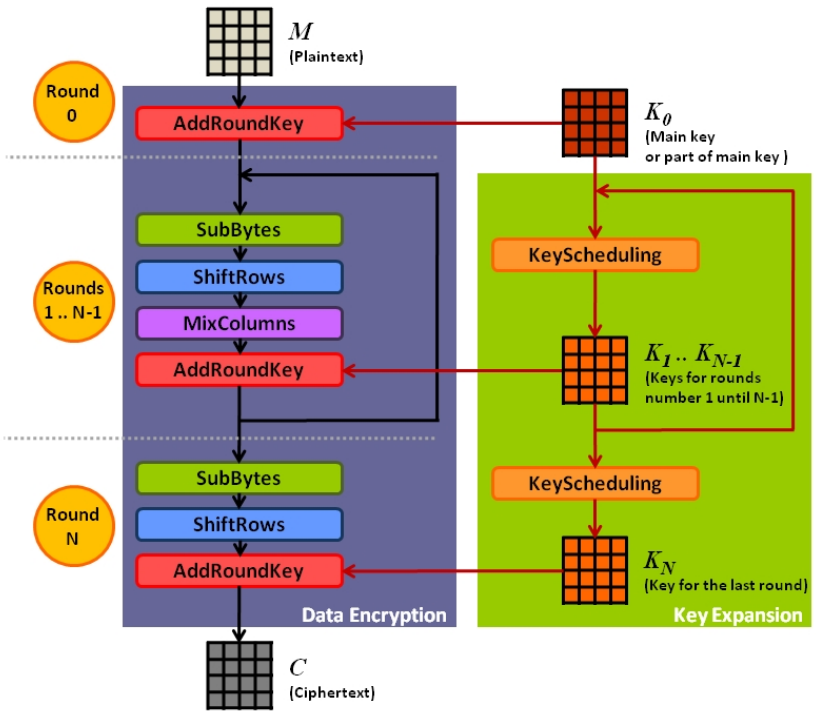


Figure 1.4: AES general outline.

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 separated 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 and the last ones, includes 4 steps: At the beginning, it exchanges the value of each matrix element, i.e. one byte value, by 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 of the same column with a different coefficient of 1, 2 or 3 for each element (`MixColumns` or `MC`) under the specific rules of 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 commercials purposes. Therefore, it has opened a new and large domain of research on 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 the researchers. They can be categorized under DFA, SEA and RR groups. Depending to the theoretical model and the physical implementation, the target can be the 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 selected kind of fault model, such as flipping a bit value, stuck the bit value at 0 or 1 or injecting a random faulty value on a byte content. Table 1.1 shows a brief state-of-the-art of 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 attacks requires only one bit difference, and so a subset of 8 options.

Besides, 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 more detailed about 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 ($100 \sim 400nm$), visible colors ($400 \sim 700nm$) and near infra-red ($700 \sim 1400nm$) and infra-red domains ($1400nm \sim 1mm$).

Laser impacts on combinational logic circuits 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.

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

Title	Type	Target	Fault Model	Foca-lization	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]

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 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, an SRAM cell contains six MOS.

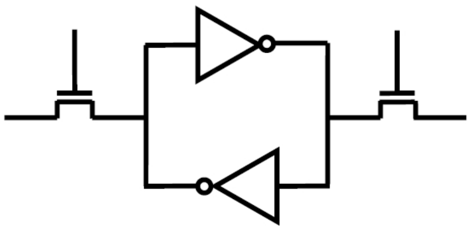


Figure 1.5: Architecture of a typical SRAM cell.

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 moment[1], the target's clock frequency, V_{cc} and temperature. Finally, laser attacks may indifferently target the chip's front side or backside.

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

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

1.3.3.1 Front side attacks

The laser attacks on front side decapsulated chips are particularly suited to green wavelength (∼ 532nm). The visibility of chips components makes positioning very easy in comparison to backside attacks. But, because of the metallic interconnects' reflective effect, it is difficult to target a component with enough 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. All in all, 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. According to this property, the backside laser attacks are more efficient as the infrared rays (∼ 1064nm) as enter deeply into the silicon and target components backwards. Positioning is more difficult for lack of visibility. Nonetheless, backside attacks allow to circumvent the reflective problem of metallic surfaces.

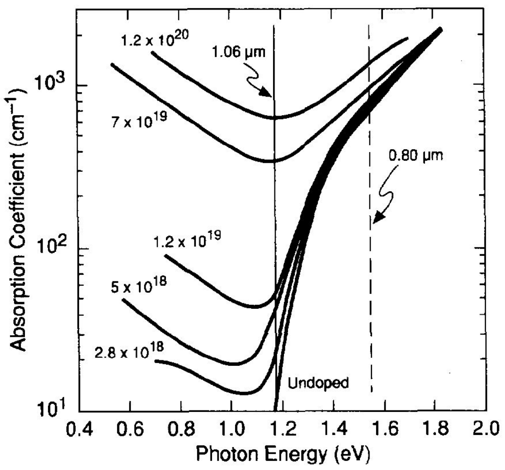


Figure 1.6: Absorption coefficient for silicon at various doping levels of p-type material.

The figure 1.6 shows 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 and 0.80 μm respectively.

According to figure 1.6 and [Melinger 1994], an infrared laser, a *e.g.* $\sim 1064\ \mu$m wavelength has a deep penetration in 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 μm are suitable as they don't entry deeply to the silicon. It includes a $\sim 532\ \mu$m green laser wavelength .

Before performing the laser fault injection, the target circuit must be prepared for such attacks. Depending to the equipments and attack's requirements, the front side or the backside of the circuit can be decapsulated.

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

-

Chapter 2

Security Characterization

Contents

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 since the beginning of prototyping phase and during 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 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 between academics and industrials in PACA region, France[1]. Its main aim is to provide security characterization facilities for testing materials during R&D projects execution from the design phase until the product validation phase.

This laboratory is currently composed of five units: *Laser fault injection*, *Electrical tests*, SPA/DPA, EMA and *Contactless* characterization benches. Installation

[1]http://www.arcsis.org

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 equipments. The laser bench, shown in figures 8.4 and 8.5, is equipped with a YAG[2] 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.

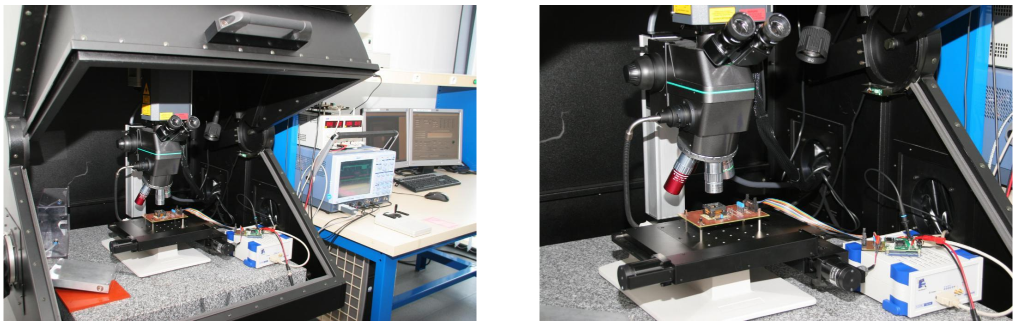


Figure 2.1: MicroPackS laserbench.

The nominal spot's size is rectangular and controlled by a rectangular shutter opening. Each two parallel side of the shutter can be opened between 0 and 2500μm. As the beam passes through a choice of Mitutoyo lenses, it gets reduced by the lens' zoom factor and loses a big 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-I [3] program. It is a running GUI[4] 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 get and save responses and curves. 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.

The laser pulse duration is about 5 ns. A waiting time of about 200 μs with a tolerance of 500 ns is involved for laser internal triggering, switching and heating be-

[2]Yttrium Aluminium Garnet or $Y_3Al_5O_{12}$
[3]**S**ecured **M**ulti-ch**ar**acterisation **T**est **I**nterface
[4]Graphical User Interface

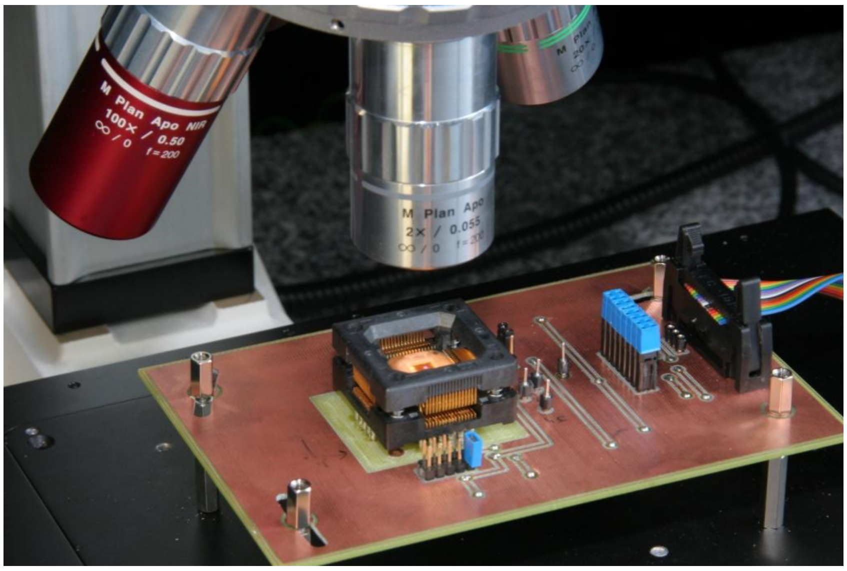


Figure 2.2: Close-up on our circuit at the laserbench.

tween 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 μm 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 permits changing the chips for different tests. A hole at the center of chip support 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 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 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 optional 5-th byte header and optional data. The header containing 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 7816-3. Figure 2.3 shows the structure of a command in T=0 protocol, according to [Rankl 2003].

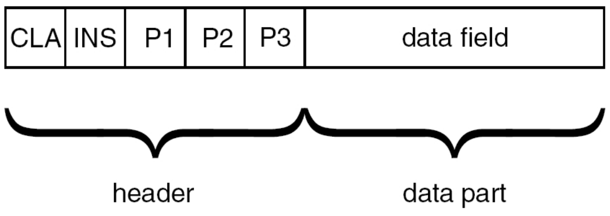


Figure 2.3: The structure of a command in T=0 protocol.

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

This microcontroller communicates at 3.59 MHz frequency with the card reader. In SOSSE environment, similarly to smart cards ISO/IEC 7816 standards, the commands are sent as 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 `Read_RAM`, `0A`

`00` sets the start address and `10` determines the length of requested bytes in hexadecimal.

```
        Fct :  Cmd T0 Cmd :  80 0C 0A 00 10
```

Then, the 16 bytes in the response are the requested data and 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 T0 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`: 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, *e.g.* `10` for 16 bytes and the plaintext. For example, the following command sends 16 bytes as a plaintext equal to `32 43 .. 34` to be encrypted by AES:

```
        Fct :  Cmd T0 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 to a second command for reading the ciphertext.

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

```
        Fct :  Cmd T0 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 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 and 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 derived and stored in the microcontroller's SRAM. The S-Box look-up table also is included in the program code and after each reset is copied to SRAM. The encryption process refers 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 is selectable to encrypt by the chosen AES-128, 192 or 256 bit versions in 10, 12 or 14 rounds respectively. In our experiments reported in this thesis, we used only AES-128 version.

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

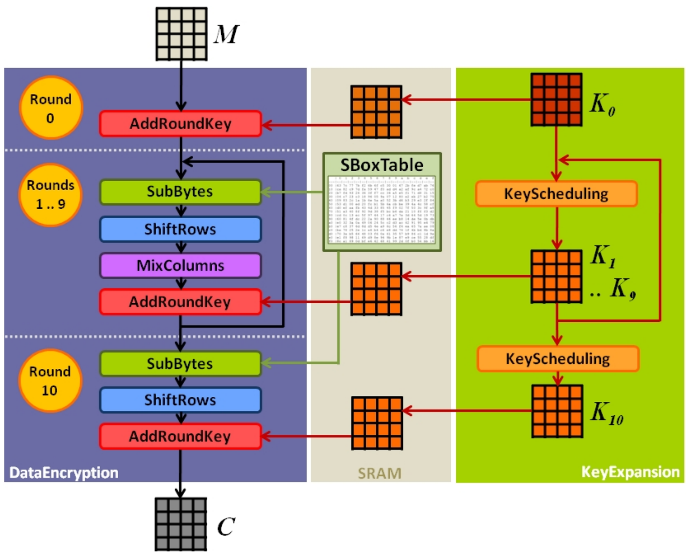


Figure 2.4: An overview of our AES implementation.

For our tests requirements, we applied often 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 “our chip”, “circuit” or “microcontroller” terms to refer to this aforementioned microcontroller used for almost the experiments, except when another circuit is mentioned.

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

2.2.2 Circuit Decapsulation

Before performing the laser fault injection, the target circuit must be prepared for such experiment. Depending to the equipments and attack's requirements, the front side or the backside of the circuit can be decapsulated.

2.2.2.1 Front Side Decapsulation

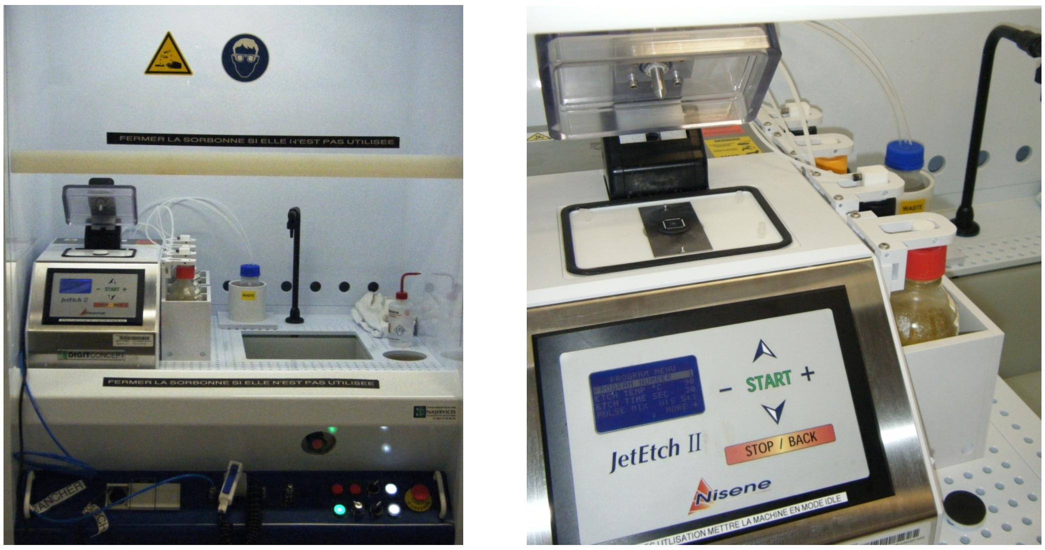

FERMER LA SORBONNE SI ELLE N'EST PAS UTILISEE
FERMER LA SORBONNE SI ELLE N'EST PAS UTILISEE
– START +
STOP / BACK
JetEtch II
Nisene

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 desired temperature between 20 and 250°C.

The optimal package opening of modern chips usually requires the use of mixed etchants. For 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 preserve structural components, but also to maintain electrical functionality [Nisene].

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

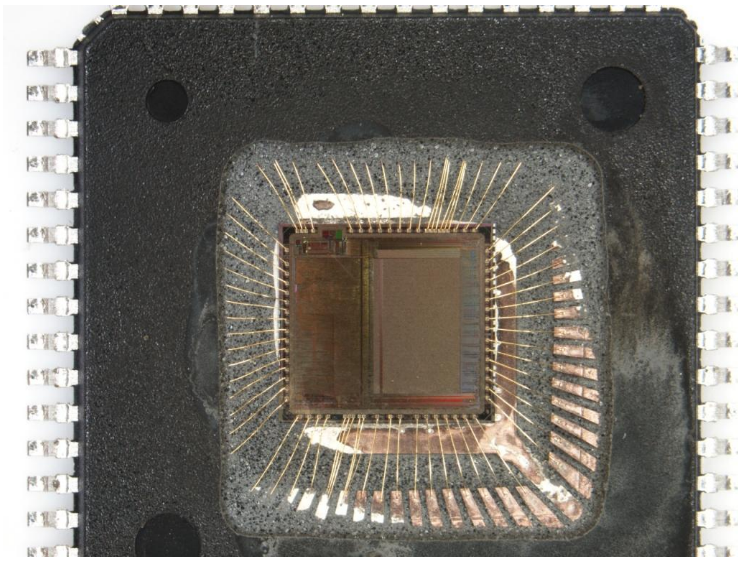


Figure 2.6: A front side decapsulated sample of our chip by chemical etching.

2.2.2.2 Backside Decapsulation

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

The backside decapsulation bench at MicroPackS laboratory is equipped by an Ultra Tec ASAP[5]-1 decapsulator unit as shown on the figures 2.7 for backside chip decapsulation. This automated mechanical machine is formed by a rotating part for 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 to its mounted tool can scratch or polish the backside surface. By using different tools in conjunction with loose or fixed abrasives, we obtain a mirror-like surface without any scratch.

Contrary to the chemical front side automated etching that takes several ten seconds or few minutes from the beginning to the end, the mechanical decapsulation requires different steps and takes several ten minutes or few hours.

[5]Automated Selected Area Preparation system

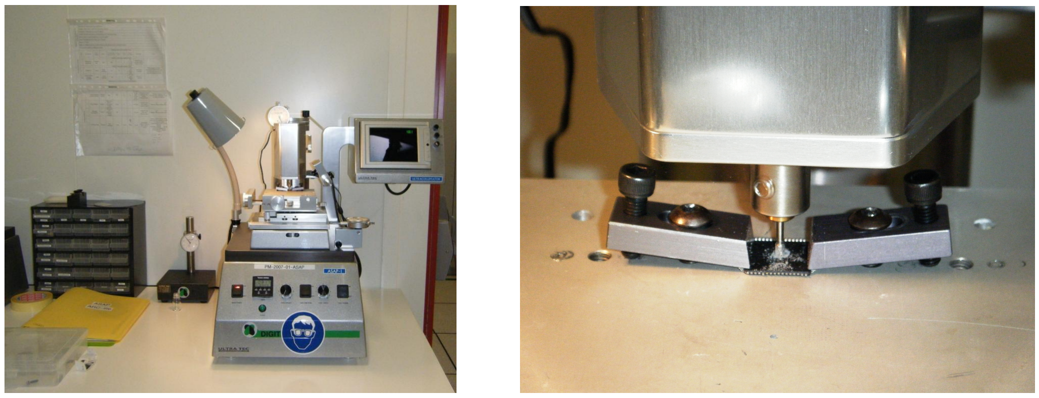


Figure 2.7: Mechanical etching bench in MicroPackS laboratory equipped with an Ultra Tec ASAP-1 decapsulator.

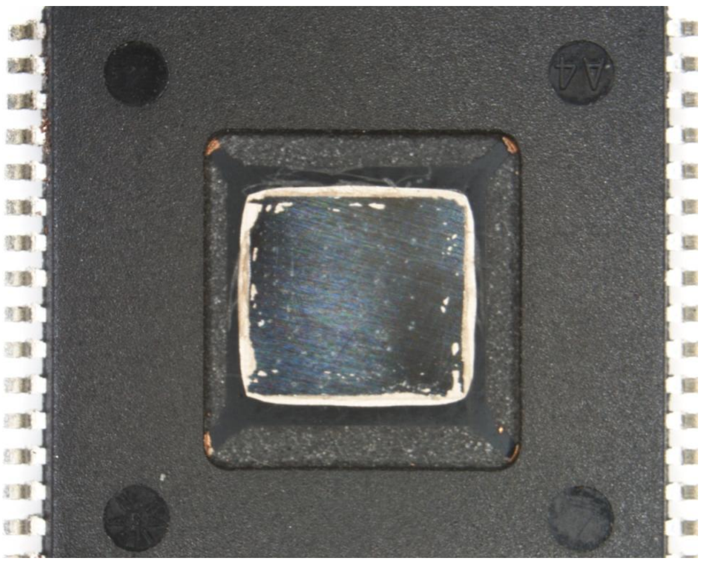


Figure 2.8: A backside decapsulated sample of our chip by mechanical etching.

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

In this section, we described chip decapsulation and preparation for laser fault injection. On the next chapter, we will show our practical results for fault attacks on an embedded AES in a decapsulated microcontroller.

2.3 Security Characterization: First Mapping of Fault Injection Susceptibility

Before starting fault attacks on AES, a good familiarity with our circuit and impacts of laser emission was necessary.

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

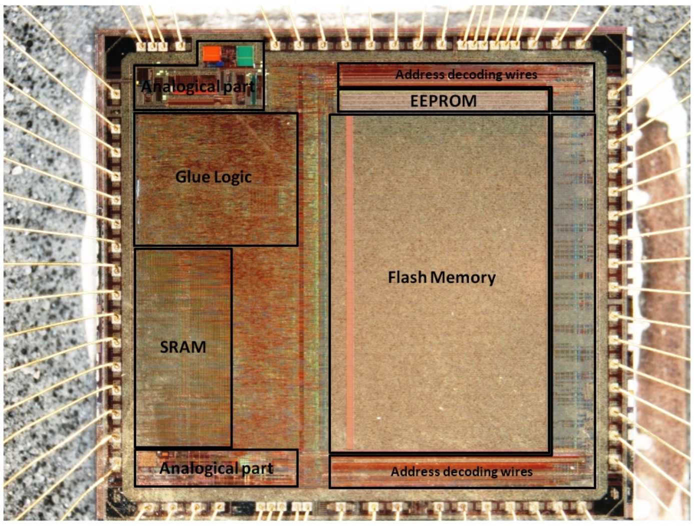


Figure 2.9: Main identified components on the original top metal overview photo.

The Picture 8.7 shows the mapping of the main functional blocs of the chip, *i.e.* SRAM, flash, EEPROM, analogical parts and logic of our circuit. As in configuration many variables are stored in the SRAM, 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 to measure the effects of laser impact on our circuit.

In our circuit and AES implementation, the SRAM has an important role as 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 perfect target for laser fault injection. Therefore, we studied our circuit's SRAM with particular attentions.

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. Write SRAM cells with a fixed value, for instance `0x00`.
2. Perform a laser emission.
3. Read SRAM cells. Find all the differences between the current and the written value. Allocate the address of changed bytes to the spatial coordinates of the scanned area.
4. Go to the next scanning area and repeat 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 are stored in SRAM. So, this cartography is very useful for our upcoming experiments.

2.3.1 Laser Spot Parameters and Fault Injection

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

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

Energy	180 pJ	1.8 nJ	9 nJ	18 nJ
Average fault number	14.3	60.8	80.4	87.3

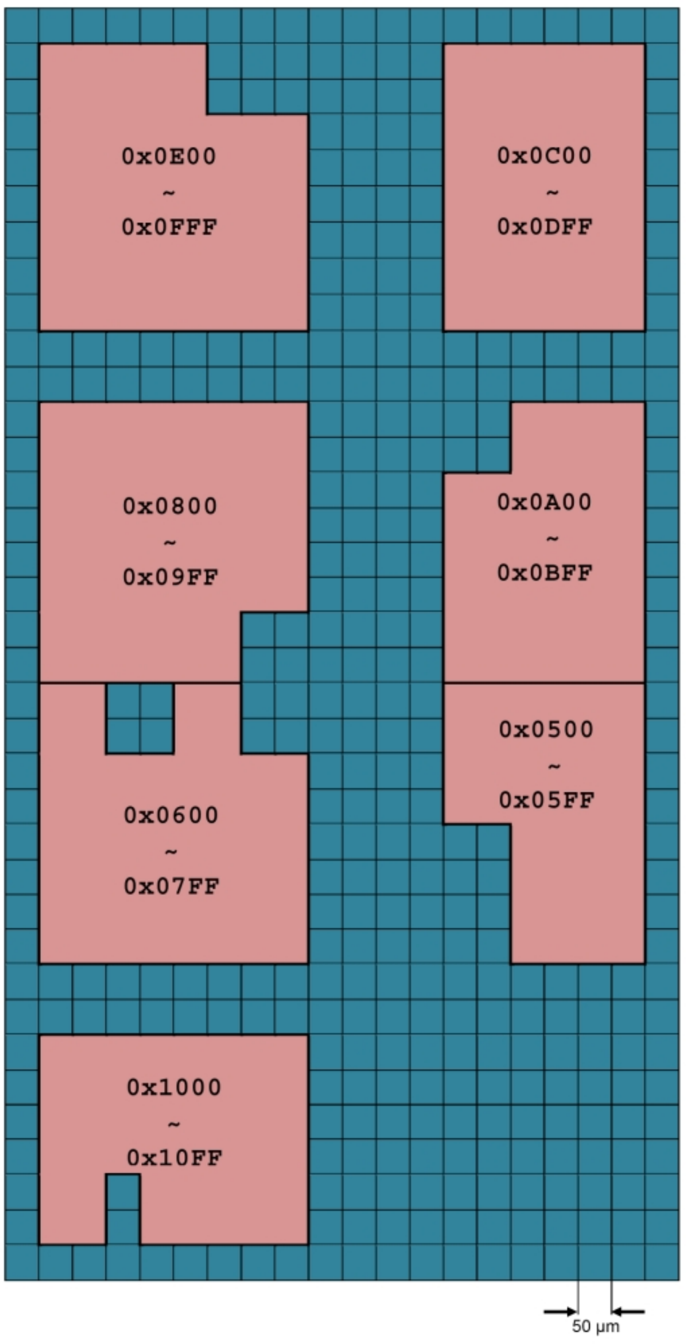


Figure 2.10: Physical allocation of SRAM's bytes on front side of our decapsulated microcontroller, discovered by laser fault injection.

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 while the beam width is smaller than $93\mu m$. In addition, the frequency of double-bit and triple-bit faults remains very low between faulty bytes.

Our results show that the chance of injecting faults on same-value bits of several bytes is much more than other bits of same bytes. It means that in several physical implementations of SRAM (*e.g.* in our microcontroller or also on 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, usually the distance of two bit cells of same value in a block of bytes (*e.g.* 256 bytes) is much closer than the distance of a bit with its neighbor 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 analyzing the circuit behavior.

Our experiments are done twice: At the first experiment, we read all the SRAM content before and after laser emissions to detect any changes on it. We performed also 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 can classify the effects of laser emission on our chip in 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, calculated values or during communications. 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 to the first reader command after laser emission, but it recovers its correct functioning very shortly.

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

```
          Fct :  Cmd TO 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 sends correctly a status equal to "61 00" correctly to the first command. Therefore, its answer is null to the second command with a status equal to "6D 00" that means *"command not supported"*. But the circuit recovers its correct functioning and answers correctly to 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 to 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.

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

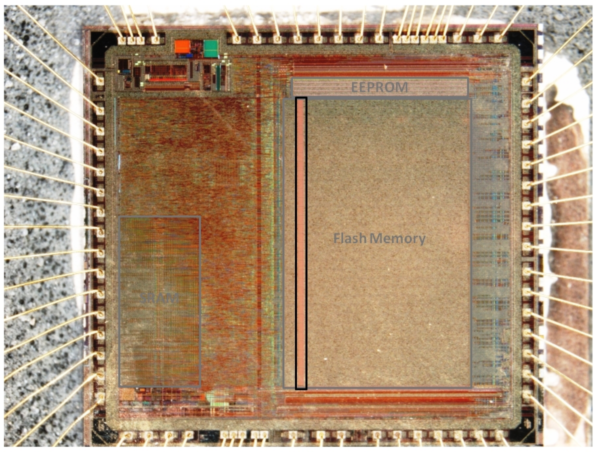


Figure 2.11: The vertical wire lines in middle of our circuit.

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

At next step, we tried to discover the reason of 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 `SubBytes` look-up table must be used during the encryption. For validating that the change is not come from faulty flash memory contents, we added a SOSSE command for changing only one byte value on `SubBytes` look-up table in SRAM. We changed this value according to the used `SubBytes` look-up table values for our chosen plaintext and embedded key at 10-th round. We found that it results in a faulty ciphertext and when we rewrite the original value, the next encryption is done correctly.

In addition, we surveyed on 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 data bus.

2.3.3 Conclusions

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

CHAPTER 3

Practical DFA by Laser on the AES

Contents

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 also, we proved the possibility of injecting laser faults within our circuit. We examined that it is possible to change some byte values by laser emission on the SRAM. Now, the main problematic is to survey if appropriated faults may be injected by laser in order to break an AES cryptosystem. In this chapter, our aim is to investigate if we can produce 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 to single-byte or single-bit fault models. The minimal diameter of a laser spot cannot successfully decrease to smaller than ∅1μm due to optical diffraction reasons. Besides, as the technology advances, the number of transistors grows on the incident area of a ∅1μm spot

Figure 3.1 shows a comparison between ∅1 and 10μm laser spot and an SRAM cell in different technology scaling. It also shows that a ∅1μm spot may have a bigger effective area on the chip that depends to the laser energy level. This minimal beam hits several transistors on the most advanced technologies and cannot physically limited to target a single-bit/byte. Besides, a laser equipment providing with ∅1μm spot is very expensive and not accessible to most of opponents. So, injecting single-bit/byte faults or even few bytes faults needs more accurate equipments and becomes less feasible by cheap laser facilities.

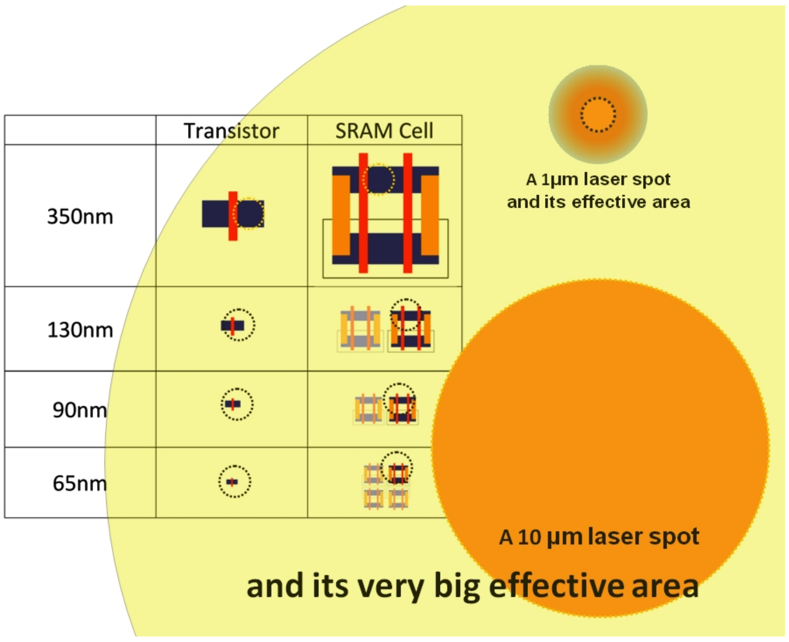

Transistor
SRAM Cell
350nm
130nm
90nm
65nm
A 1µm laser spot and its effective area
A 10 µm laser spot
and its very big effective area

Figure 3.1: 1 μm & 10 μm laser spot diameters vs technology scaling.

In the following sections of this chapter, we describe two classical DFA methods on AES and our experiments for performing required single-bit or single-byte laser fault injection. We survey if we can inject required faults by our laser equipment on our circuit.

3.2 Piret-Quisquater's Differential Fault Analysis

3.2.1 Attack's Scheme

Gilles Piret and Jean-Jacques Quisquater presented in [Piret 2003] a theoretical DFA attack on AES. 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 to be successful.

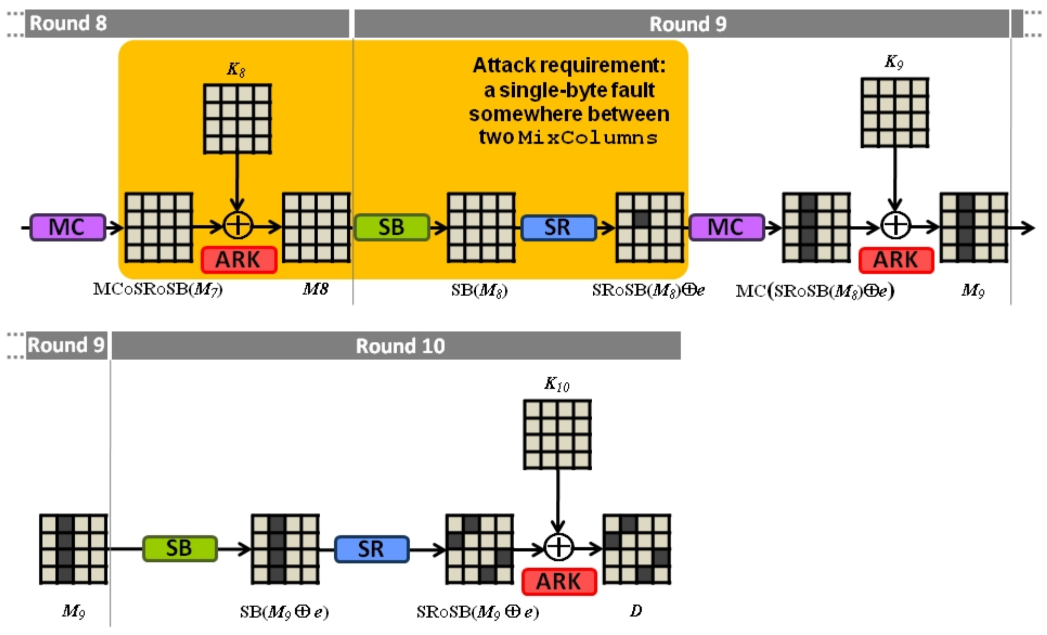


Figure 3.2: Propagation of a single-byte fault at `MixColumns` input of the round 9.

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 as the attack's requirement. Figure 3.2 also presents the fault propagation and diffusion into four bytes. Then, the attack scheme described in [Piret 2003] allows 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 survey how we can implement this attack on our circuit embedding an AES. We described in section 2.3 that in our circuit the round keys are derived and stored in the microcontroller's SRAM. As it is very difficult to target the chip's ALU[1] and inject only a single-byte fault during a very specific instant between the end of `MixColumns` of the 8-th round and before the `MixColumns` of the 9-th round, targeting K_8 is a proper option for our attack.

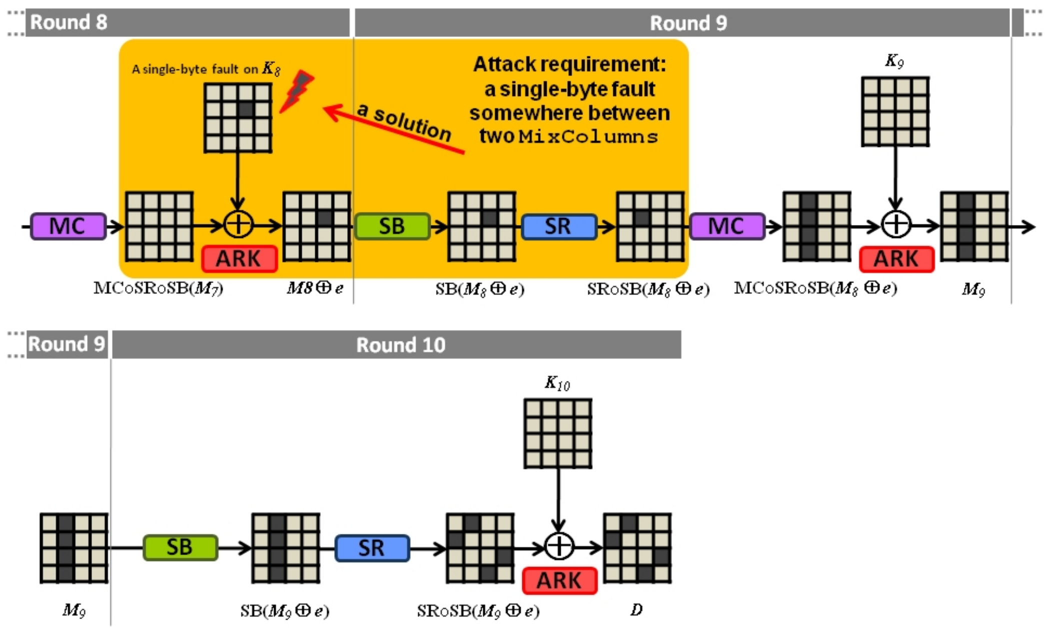


Figure 3.3: Effects of a single-byte fault at `SubBytes` input of the round 9.

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's scheme allows 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 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

[1] Arithmetic Logic Unit

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 steps 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. **Fining spatial coordinates:** We tried to maintain fault injection on a single-byte of K_8, but to exclude any fault on the following round keys, *i.e.* K_9 and K_{10}. We performed micrometer displacements of 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 use of K_8 and before use of K_9. Therefore, the first encryption will be correct and the following ones become faulty until the next circuit reset.

3. **Temporal positioning:** We applied a temporal accuracy to the laser emission time, in a short interval time after `AddRoundKey` of 7-th round and before `AddRoundKey` of the 8-th round. Consequently, the logical effect of any fault on the previous round keys were discarded, despite the fact that they were injected and physically existent on the SRAM. So, the single-byte fault on K_8 were the only faulty round key byte that is 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 to describe 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 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 with a $20\times$ lens, a green laser spot of 3.75×3.75 μm and $\simeq 15pJ$ per shot (at laser emitter exit) on our chip's front side.

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 the fault injection to corrupt only 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.

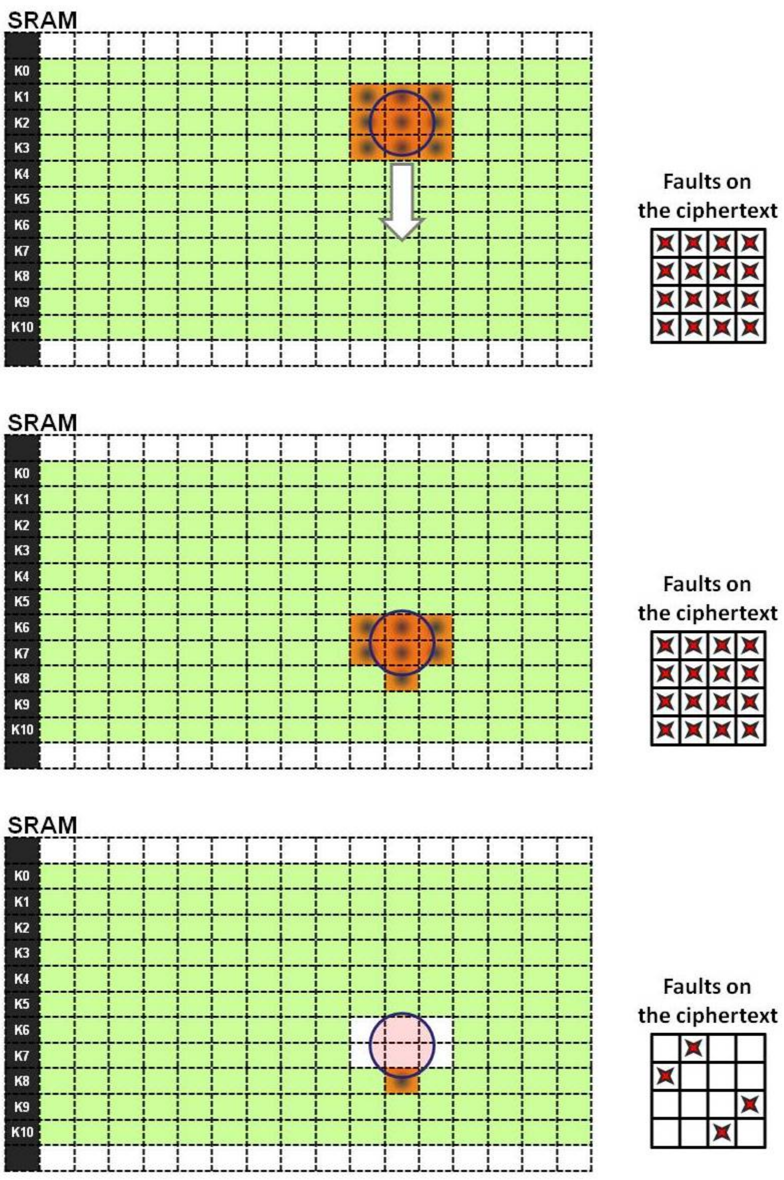


Figure 3.4: Exploration process of our experiment are shown in three steps. First, we search for K_8 bytes in SRAM. Then, we displace the spot on SRAM in order to target only one K_8 byte and nothing on the following round keys. Finally, by time tuning on t_8, we inject logically only a single-byte fault on K_8.

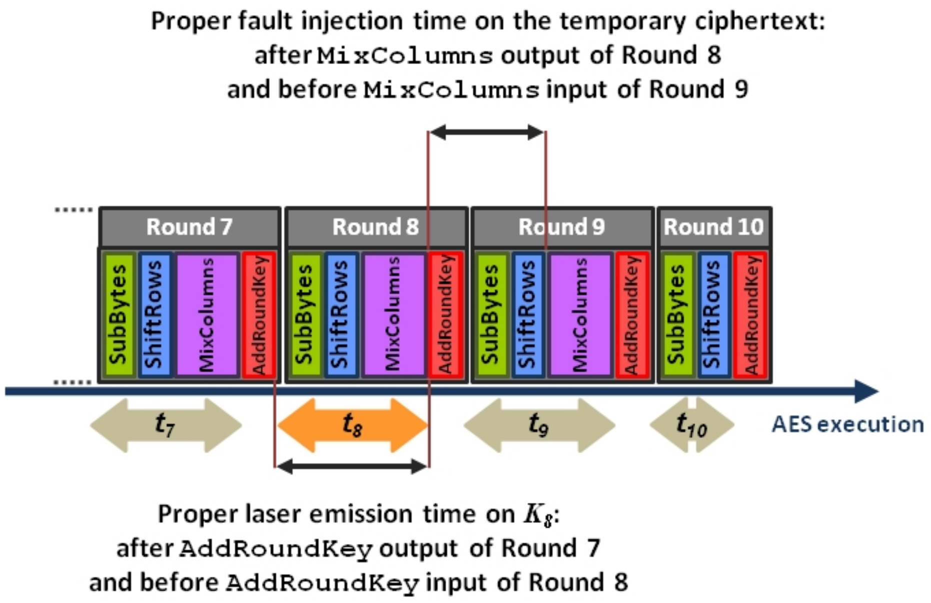

Proper fault injection time on the temporary ciphertext:
after MixColumns output of Round 8
and before MixColumns input of Round 9
Round 7
Round 8
Round 9
Round 10
SubBytes
ShiftRows
MixColumns
AddRoundKey
t_7
t_8
t_9
t_{10}
AES execution
Proper laser emission time on K_8:
after AddRoundKey output of Round 7
and before AddRoundKey input of Round 8

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 affect only logically the oncoming round keys. Even, if they change previous round keys, their faulty values don't participate to the encryption, despite the fact that they physically exist. Hereafter, we call each of these time intervals to its respective round; for instance t_8 for the time interval for injecting faults between 7-th round and 8-th 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 perform successfully Piret-Quisquater's DFA in our circuit by a laser beam that hits several bytes. It concludes that careful spatial and temporal coordination may allow to deceive the encryption process to consider logically 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 it still possible to achieve this fault injection successfully without memory access?

3.2.3 Practical Experiment Without Memory Access

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

For this new experiment, we searched blindly 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 has not any fault. Contrary, 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 previous round keys passes at least two times into consecutive `ShiftRows` and `MixColumns` operations that amplify the fault and so change all the bytes of the ciphertext. Figure 3.6 shows the consequences of one single-byte fault injected on any round key.

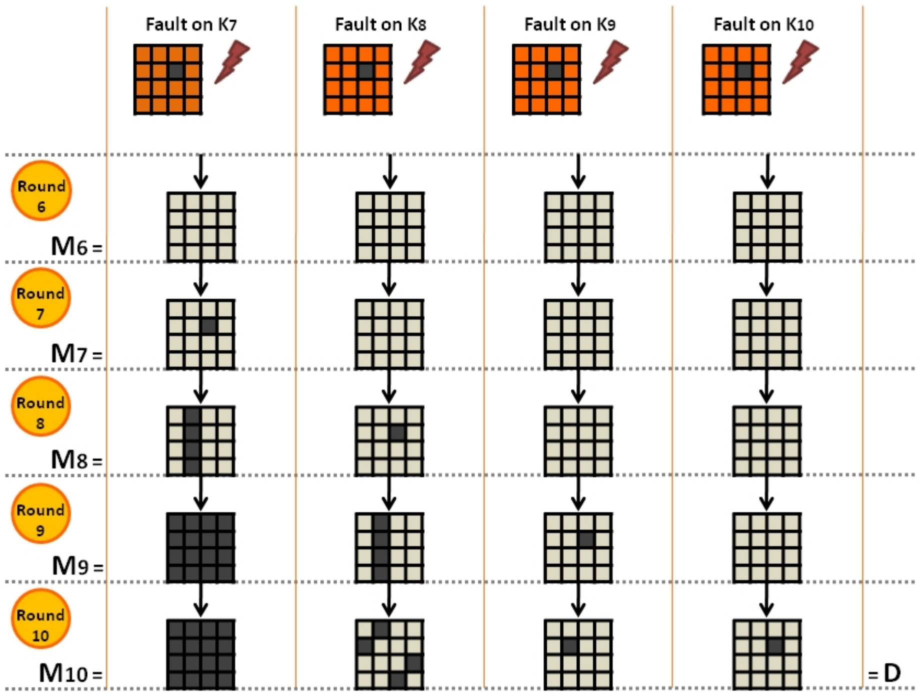


Figure 3.6: Effects of `MixColumns` on fault propagation for one faulty single-bit/byte round key on the temporary ciphertext at the end of each round and at the end of algorithm.

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

In the experiments with memory access, we applied the temporal accuracy at 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

Table 3.1: Potential faulty K_is as function of observed faulty ciphertext bytes.

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

logical effects of *"early faults"* occurring in any K_i preceding K_8. Top part of the figure 3.7 shows this exploration.

Then, we searched for spatial accuracy to target only the first single-byte on K_8, as shown on bottom part of the 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 attack on AES as it requires only 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 to deceive the encryption process to consider logically 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 to modify memory cells in the absence of countermeasures.

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

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 survey the feasibility of performing a single-bit DFA on our embedded AES.

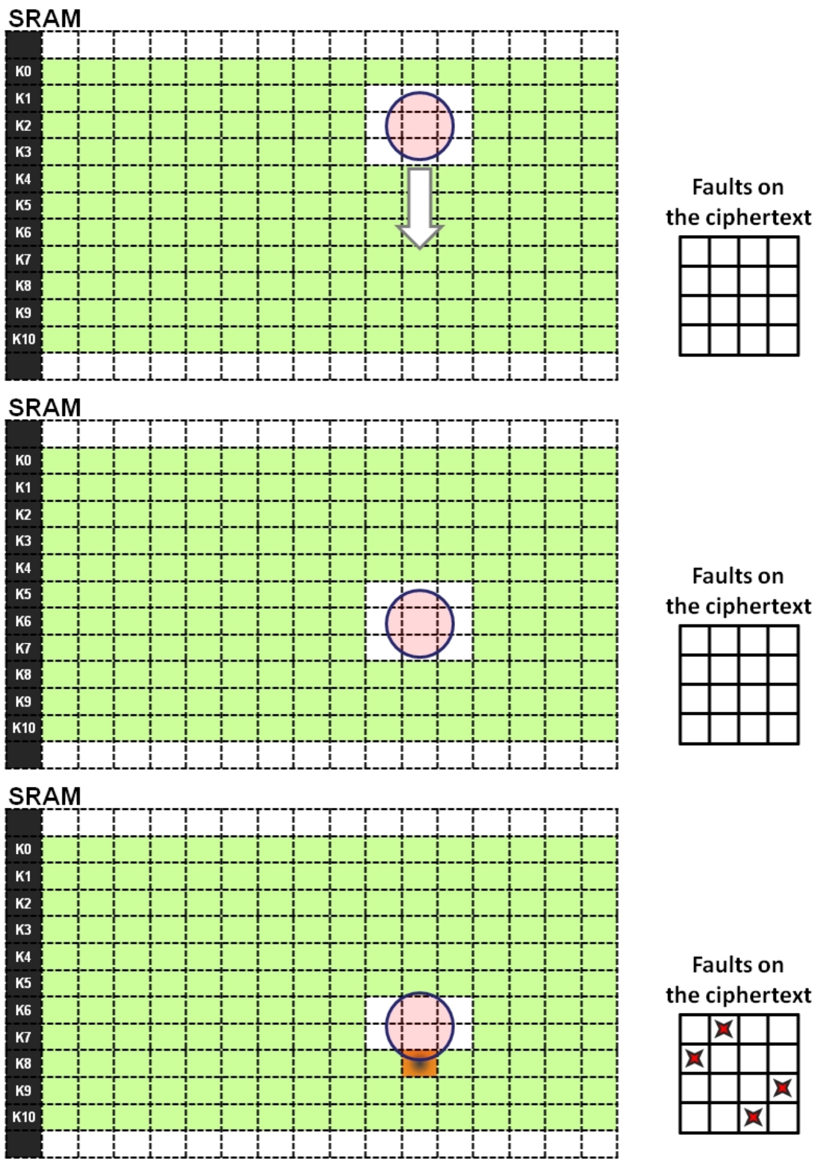


Figure 3.7: Exploration process for Piret-Quisquater's DFA by blindly experiment. The laser emission time is set to t_8 during exploration.

3.3.1 Attack's 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.

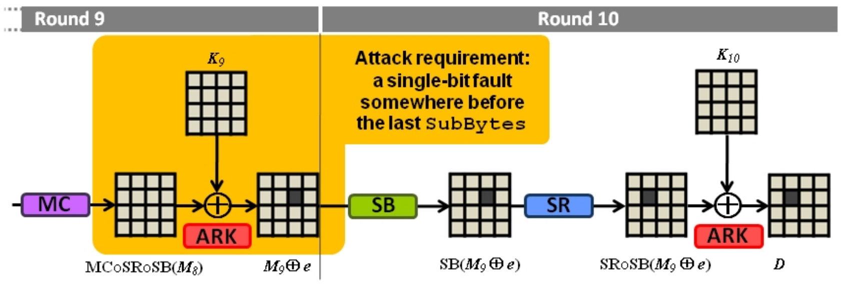


Figure 3.8: Giraud's bit DFA.

To discover one byte of K_{10}, the attack requires to repeat a single-bit fault for at least three different plaintexts. The three faulty results are then compared to their corresponding correct ciphertexts 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 M_9 corresponding byte. Equation 3.1 shows this operation:

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

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

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

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

$$D = \texttt{SB}(M_9 \oplus e) \oplus K_{10} \tag{3.3}$$

[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:

$$\Delta = \texttt{SB}(M_9 \oplus e) \oplus \texttt{SB}(M_9) \tag{3.4}$$

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 replaced for each pair of (M_9, e) values.

$$K_{10} = \texttt{SB}(M_9 \oplus e) \oplus D \tag{3.5}$$

Each pair of a faulty and its corresponding correct ciphertexts for one byte concludes in a set of hypotheses on the value of corresponding K_{10} byte. By repeating the fault injection for at least three different plaintexts, the opponent creates the same number of hypothesis sets on the corresponding K_{10} byte value. Then, sets are intersected to spot the single hypothesis that reveals 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 an unique value for a K_{10} byte.

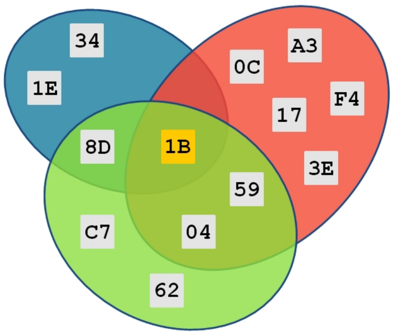


Figure 3.9: An intersection between three hypothesis sets for a K_{10} byte value.

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 fault is not single-bit. However, the opponent may perform a union operation between the sets instead of intersection operation. The incorrect hypotheses will be discarded easily 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.

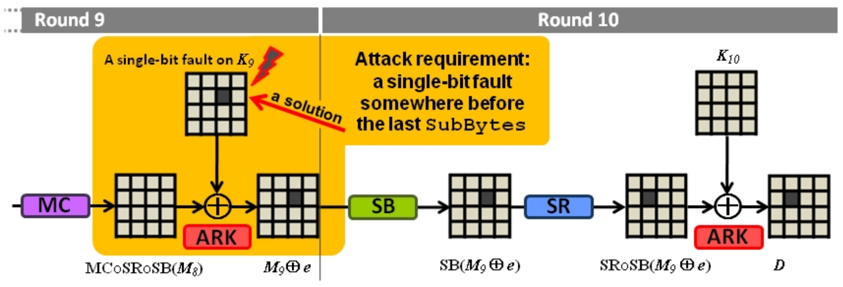


Figure 3.10: Giraud's bit DFA.

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 used again the three steps solution, with a slight change: We performed temporal positioning at the first step. It reduces the number of faulty bytes on the final ciphertext and so it was easier to fine spatial coordinates to inject a single-bit fault on K_9 with both options of memory access or blindly tests.

Figure 3.11 demonstrates our spatial explorations. As shown on top part 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 only on K_9 bytes and protect K_{10} from any fault. 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 to the encryption.

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

3.3.3 Conclusion

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

According to the results of two previous sections for implementing Piret-Quisquater and Giraud's singe-bit DFA, even when targeting a single-bit or single-byte is physically impossible because the beam hits a few other bytes, careful spatial

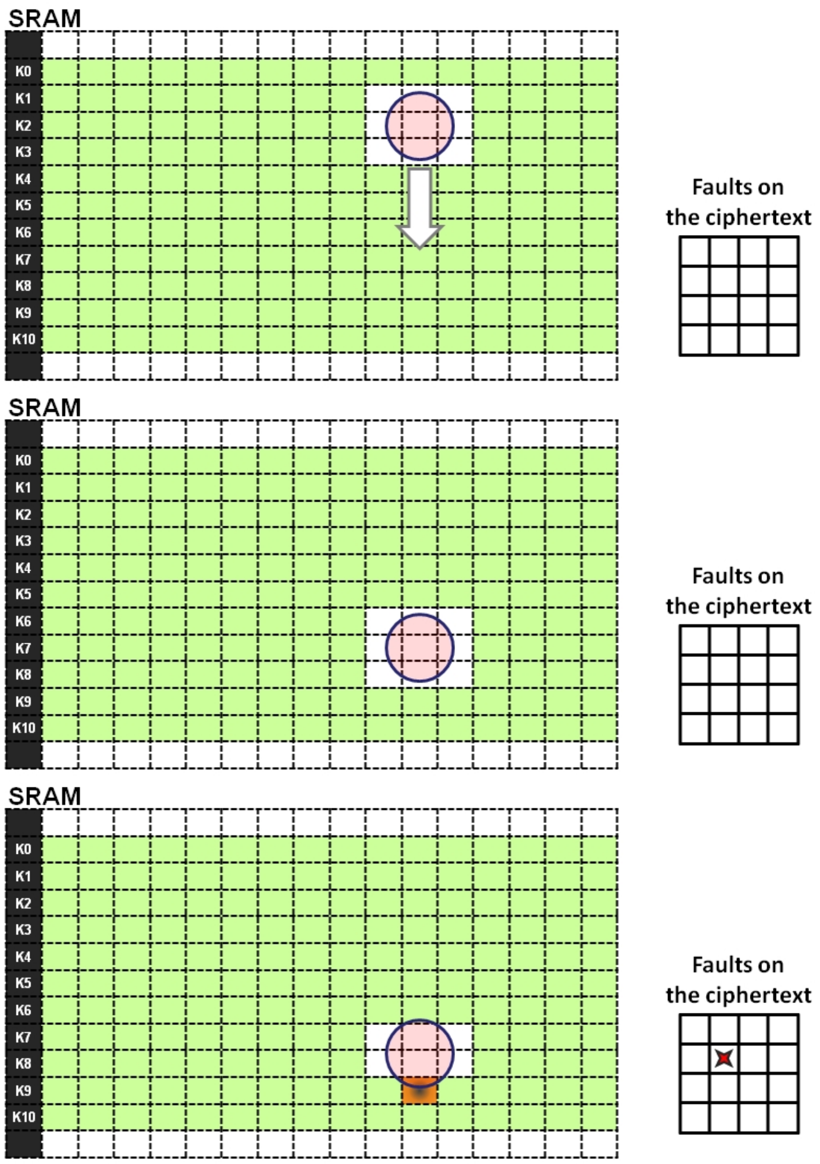


Figure 3.11: Exploration process for Giraud's Single-Bit DFA. The laser emission time is set to t_9 during exploration.

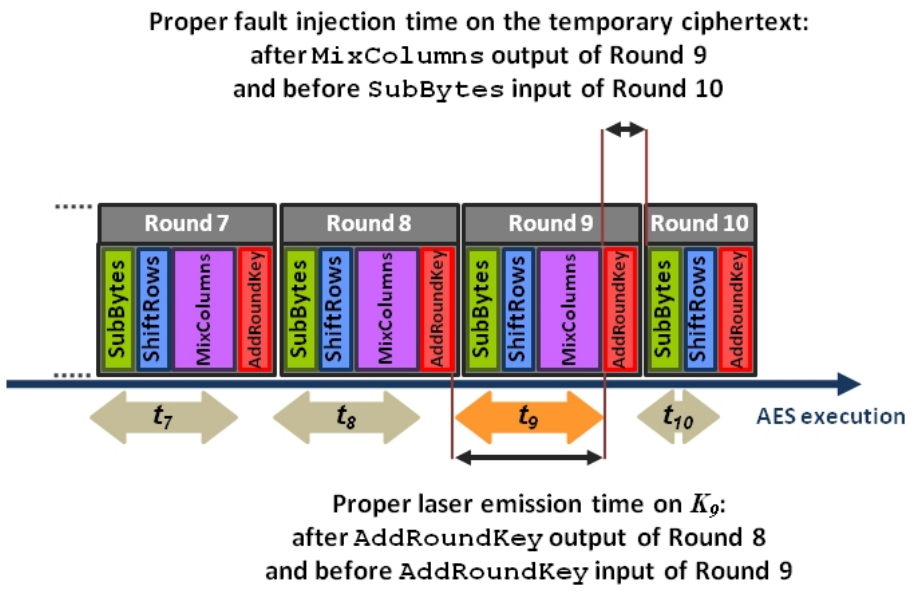

Proper fault injection time on the temporary ciphertext:
after MixColumns output of Round 9
and before SubBytes input of Round 10
Round 7
Round 8
Round 9
Round 10
SubBytes
ShiftRows
MixColumns
AddRoundKey
t_7
t_8
t_9
t_{10}
AES execution
Proper laser emission time on K_9:
after AddRoundKey output of Round 8
and before AddRoundKey input of Round 9

Figure 3.12: Giraud's bit attack timing.

and temporal coordination may allow to exclude extra faults and deceive the encryption process to consider logically only the required fault.

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

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 accuracy on the attack time, 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 key and the opponent cannot reduce the number of faulty bytes. Then, we discuss about the possibility of performing bit-level DFA.

3.4.1 1-st 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 a 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 of writing and reading a part of the memory confirmed this guess during the preliminary tests reported in section 2.3.

In implementation of our microcontroller, the distance of two bit cells of 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 for performing successfully Giraud's bit DFA in the subsection 3.3.2.

On the other hand, after `MixColumns` exit of the 9-th round, there is no more `MixColumns` step 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's scheme and even speeds up the DFA.

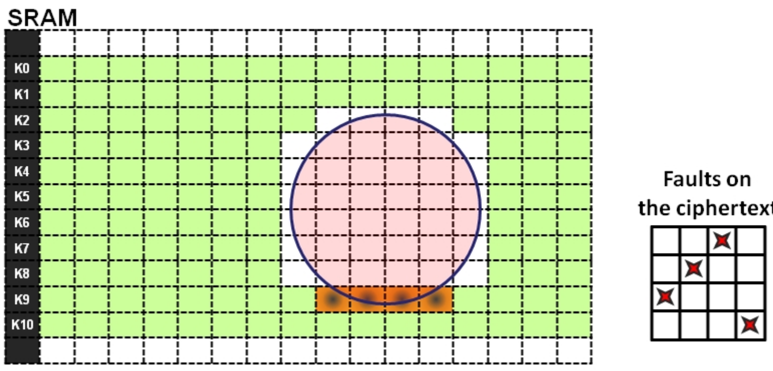


Figure 3.13: Exploration process with a big laser spot. The laser emission time is set to t_9 during exploration.

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.

3.4.2 2-nd Scheme for the Extended Attack

Figure 3.13 showed how the opponent can limit fault injection to several byte 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 protect completely K_{10} from fault injection due to the large size of a laser spot. Therefore, few single-bit faults are injected on K_{10} byte. 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:

$$D = \texttt{SB}(M_9) \oplus (K_{10} \oplus e) \tag{3.6}$$

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}:

$$\Delta = C \oplus D = e \tag{3.7}$$

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}, 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 fault refers to all the faulty bytes with more than one bit differences in comparison to their correct value. Although, the class of K_{10}-related fault contains all the faulty bytes with only one bit difference. As the contents of this class don't correspond to Giraud's bit fault model, they must be excluded from list of faulty bytes for the DFA.

3.4.3 Practical Experiment

The 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 been 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 on bottom part of figure 3.14.

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

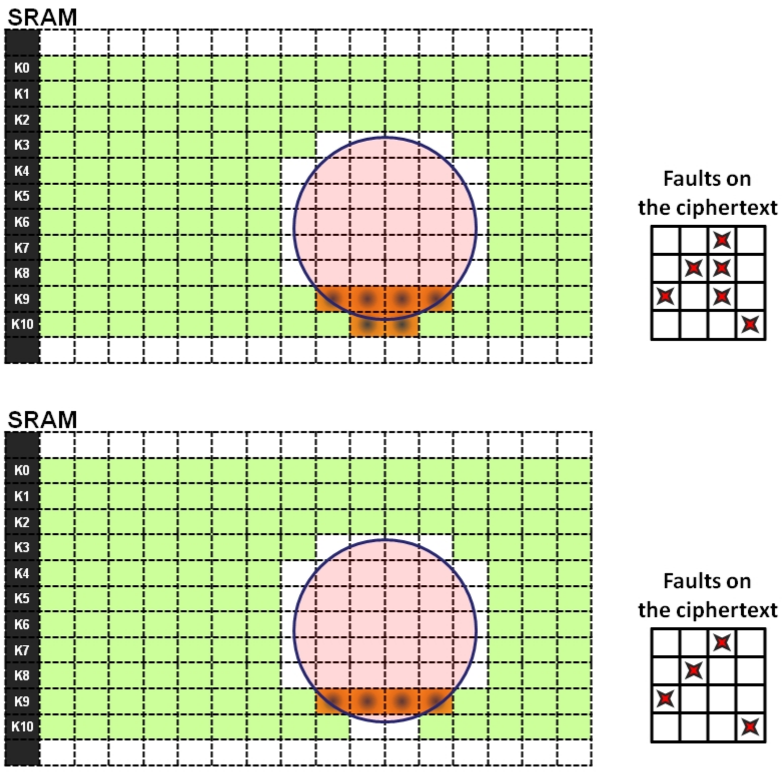
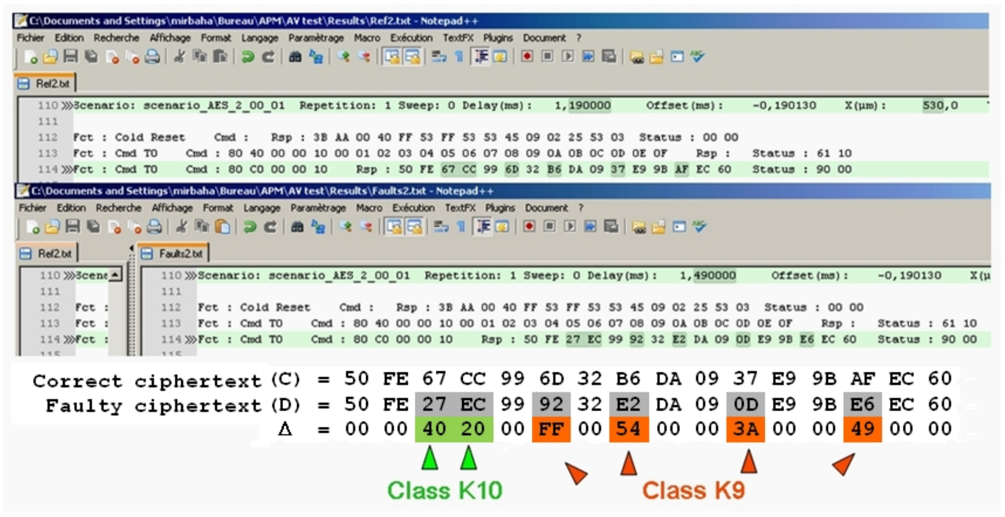


Figure 3.14: Exploration process and results classification. The laser emission time is set to t_9 during exploration. Then, the classification process excludes K_{10}-related faults.



```
C:\Documents and Settings\mirbaha\Bureau\APM\AV test\Results\Ref2.txt - Notepad++
»Scenario: scenario_AES_2_00_01  Repetition: 1 Sweep: 0 Delay(ms):   1,190000    Offset(ms):    -0,190130    X(µm):      530,0

 Fct : Cold Reset    Cmd :   Rsp : 3B AA 00 40 FF 53 FF 53 53 45 09 02 25 53 03  Status : 00 00
 Fct : Cmd TO    Cmd : 80 40 00 00 10 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F    Rsp :   Status : 61 10
»Fct : Cmd TO    Cmd : 80 C0 00 00 10      Rsp : 50 FE 67 CC 99 6D 32 B6 DA 09 37 E9 9B AF EC 60   Status : 90 00

C:\Documents and Settings\mirbaha\Bureau\APM\AV test\Results\Faults2.txt - Notepad++
»Scenario: scenario_AES_2_00_01  Repetition: 1 Sweep: 0 Delay(ms):   1,490000    Offset(ms):    -0,190130    X(µ

 Fct : Cold Reset    Cmd :   Rsp : 3B AA 00 40 FF 53 FF 53 53 45 09 02 25 53 03  Status : 00 00
 Fct : Cmd TO    Cmd : 80 40 00 00 10 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F    Rsp :   Status : 61 10
»Fct : Cmd TO    Cmd : 80 C0 00 00 10      Rsp : 50 FE 27 EC 99 92 32 E2 DA 09 0D E9 9B E6 EC 60   Status : 90 00
```

```
Correct ciphertext (C) = 50 FE 67 CC 99 6D 32 B6 DA 09 37 E9 9B AF EC 60
  Faulty ciphertext (D) = 50 FE 27 EC 99 92 32 E2 DA 09 0D E9 9B E6 EC 60
                      Δ = 00 00 40 20 00 FF 00 54 00 00 3A 00 00 49 00 00
```

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.

of K_{10} and it will be classified as a K_9-related fault. So, it creates some false assumptions on 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. Or, an exhaustive search will be needed for the remaining assumptions 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 is only 48 cases between 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 of about 2.34%, 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. Or, like the other case, an exhaustive search will be needed for the remaining assumptions to examine them and find the correct one.

3.4.4 Conclusion

In summary, we can perform successfully 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 performed successfully 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} value. Our experiments were conducted with a $20\times$ lens, a green laser spot of 4.375×5.625 μm and $\simeq 19.7$ pJ per shot (at laser emitter exit) on our chip's front side.

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

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

Therefore, Giraud's bit DFA is classed after Piret-Quisquater's method, as it 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 implementation of SRAM memories, the feasibility of injecting several single-bit faults on different bytes is more likely that the targeting of several bits of a same byte. This strength comes from the design of several memory array layout implementations that facilitates the feasibility of single-bit faults on one or few bytes (Please refer to figure 3.16, taken from [Pavlov 2008]). The hardware implementation has an important consequent 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 at 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 equipments in comparison to the fabrication technology size for injecting faults. Considering K_8, the model can be extended to cover maximum 4 distinct 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

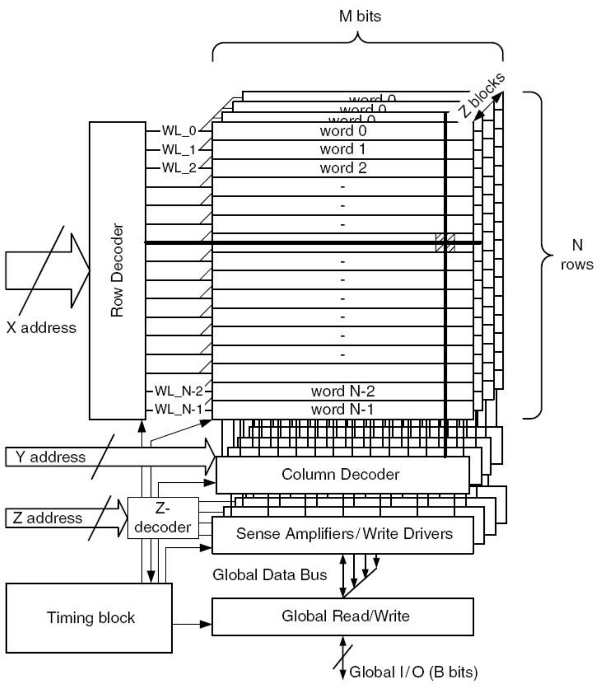


Figure 3.16: A SRAM block diagram.

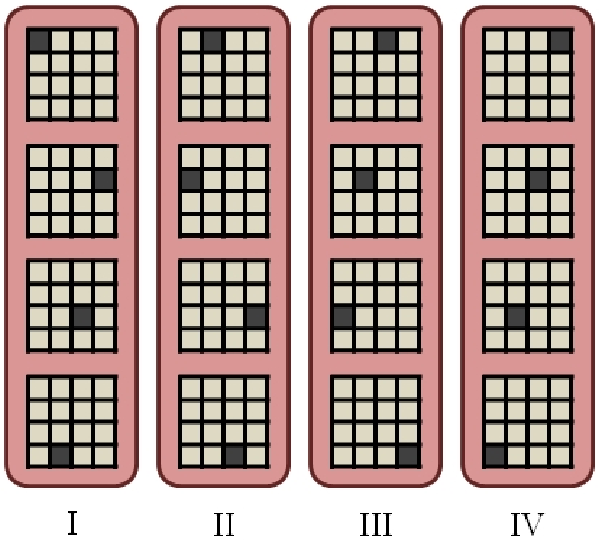


Figure 3.17: Four different sets of possible faulty bytes on K_8. When more than one faulty byte is injected from an individual set on K_8, their effects create multiple faults on the respective four bytes after `MixColumns`. Therefore, Piret-Quisquater's DFA process cannot find the key values.

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 same bytes of the temporary ciphertext. So, the faults on the final ciphertext cannot any more reveal the correct values of the key during DFA processes, as shown on figure 3.18.

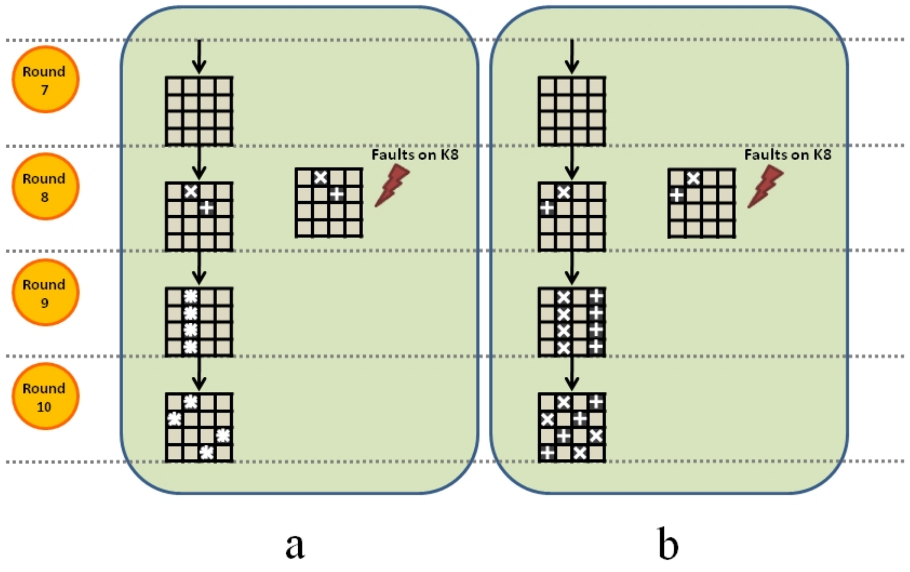


Figure 3.18: Description of additional faulty bytes on K_8: In part (a), when two faulty bytes are injected from same set on K_8, their effects change twice the content of same column after the next `MixColumns` and Piret-Quisquater's DFA process cannot find the key values. Besides, part (b) shows that two faulty bytes injected from two different sets on 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 few additional faulty bytes appears on the targeted key and/or the next ones? The solutions reported in [Dutertre 2010] seem to not be successful any more.

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 on the K_8, two different situations may happen:

- The fault is injected on one of three other bytes that participate with the targeted faulty byte on 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 discards effect of the targeted faulty byte and disrupts the DFA process.

- The fault is injected on 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 rely that each quadruple of faults is 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 happen:

- The fault is injected on one of the four corresponding bytes 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 discards effect of the required (targeted) faulty byte and disrupts the DFA process.

- The fault is injected on 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 rely that the desired quadruple of faulty bytes is 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 model can be extended to cover maximum 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 not more than one faulty byte of each set of faults exists. Otherwise, the effects of multiple faults from a single set will be repeated on same bytes of the temporary ciphertext. So, the faults on the final ciphertext cannot any more reveal the correct values of the key during DFA processes, as shown on figure 3.18.

Therefore, additional faulty bytes on the targeted round key or the next one are not corresponding to the Piret-Quisquater's DFA. This DFA model is very dependant to its required model. Besides, solutions demonstrated in 3.2.2 cannot discard the effects of undesired 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 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 on the same targeted byte: In this case, the obtained faulty result creates wrong assumptions and cannot lead to the key value. But, an additional pair of corresponding faulty and correct ciphertexts can discard 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. Because, in the case of a very limited fault injection, the chance of two single-bit faults on neighbor 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 separately for a set of bytes and with distance from the bits of other values.

- The fault is injected on 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 needed faulty ciphertexts for 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 weak. But, additional faulty bytes with single-bit faults, even until 15 more bytes, accelerate the DFA process by reducing 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 will be very probably a single-bit fault for the reasons of memory array layout design on our circuit and 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 $K9$: 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 discards the effect of targeted faulty byte and perturbs the DFA process. But, 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 on one of 15 other bytes: In this case, the additional fault can be detected immediately by the opponent and discarded from the DFA processes. Because it creates only one single-bit fault on the final ciphertext. So, it can be detected by a comparison between the correct and 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 not any limit for single-bit faults on K_9. But, the number of single-bit faults on the 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 about 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 few more bits more than the reference model may still permit to perform DFA. After finding out the good flexibility of Giraud's bit DFA, we presented our extended DFA method, based on spatial and temporal accuracies and a classification of the results. Besides, the feasibility 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 does not disturb the results.

On the other hand, Piret-Quisquater's DFA has a practical weakness that is illustrated clearly on the figure 3.18. In an experiment, the opponent performs a single-byte fault attack on K_8 and he obtains the final ciphertext with corresponding number of faults, *i.e.* 4 faulty bytes. Then, how the opponent can be sure that only one faulty byte is injected on K_8, and not an additional byte that has same effects? On the Giraud's bit model, this problem does not exist and this is a practical advantage in comparison to Piret-Quisquater's DFA. The only weakness on the feasibility 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 the Table 3.5.3. Our extended attack, based on Giraud's-bit model, shows more feasibility between them by laser attack on AES for several SRAM implementations. When few additional bytes are injected on the targeted round key and its following key, our extended Giraud's bit DFA may 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 few faults are injected on the next round keys, a classification between K_9-related and K_{10}-related faults may exclude the second class faults from the 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 more than that is usually considered on unprotected chips. Especially, the design of physical layout may improve the possibility of required faults. So, developing the proper countermeasures against laser fault attacks is necessary for security of cryptographic circuits.

The feasibility study of performing such faults by different category of opponents, regarding to their level of technical knowledge and financial budget for equipments, is necessary for developing proper countermeasures.

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

	Piret-Quisquater's DFA	Giraud's-bit DFA	Our Extended-Giraud's-bit DFA
Fault model	Single-byte	Single-bit	One or several single-bit
Target	Temporary ciphertext between `MixColumns` exit of R_8 and `SubBytes` entry of R_9	Temporary ciphertext between `MixColumns` exit of R_9 and `SubBytes` entry of R_{10}	Temporary ciphertext between `MixColumns` exit of R_9 and `SubBytes` entry of R_{10}
Alternative target on the round keys	K_8	K_9	K_9
Additional fault on the previous round keys	Disrupts the DFA process	Disrupts the DFA process	Discarded by spatial and temporal accuracies
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 wider 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 on several bytes of K_9, did not attenuate and even sped it up. Furthermore, few additional faults on K_{10} could be identified and excluded from DFA process by a simple classification. Nevertheless, when the opponent cannot protect K_{10} from numerous faults on it, our DFA will be no more 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's Scheme

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

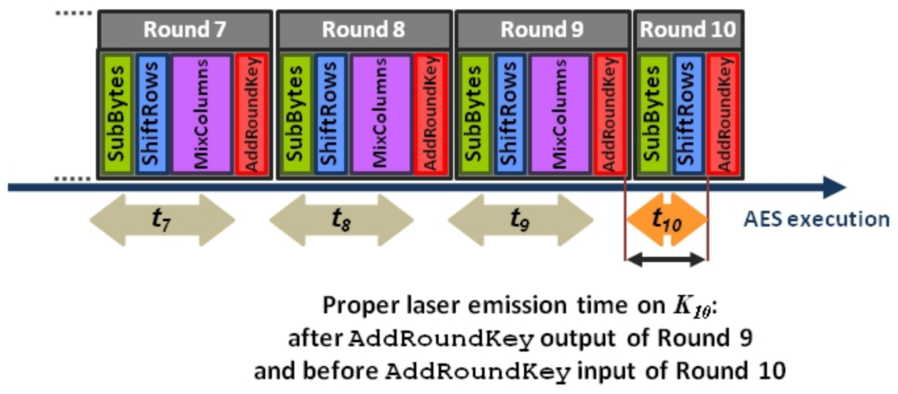

Round 7
Round 8
Round 9
Round 10
SubBytes
ShiftRows
MixColumns
AddRoundKey
t_7
t_8
t_9
t_{10}
AES execution
Proper laser emission time on K_{10}:
after AddRoundKey output of Round 9
and before AddRoundKey input of Round 10

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

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

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

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 not provide any information about the original key value:

$$\Delta = C \oplus D = e \tag{3.9}$$

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 product of two faults on corresponding K_9 and K_{10} bytes. Each pair of a faulty byte and its corresponding correct byte on the ciphertext can be expressed by the equations 3.10 and 3.11:

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

$$D = \mathtt{SR} \circ \mathtt{SB}[\mathtt{MC} \circ \mathtt{SR} \circ \mathtt{SB}(M_8) \oplus K_9'] \oplus K_{10}' \tag{3.11}$$

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

$$K_9' = K_9 \oplus E_9 \tag{3.12}$$

Therefore, we replace K_9' values in equations 3.11 according to equation 3.12 and we get new equations 3.13:

$$D = \mathtt{SR} \circ \mathtt{SB}[(M_9) \oplus E_9] \oplus K_{10}' \tag{3.13}$$

For exploiting successfully this DFA, we need at least three pairs of corresponding correct and faulty ciphertexts. We call them by 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 9-th round.

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

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

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

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

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

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

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

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

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

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

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 correct ciphertexts. So, we obtain the equations 3.18:

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

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:

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

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

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

Finally, we perform an exhaustive search between 2^8 possible values for each M_9^a byte and between 2^8 possible values for each corresponding E_9 byte. This exhaustive search leads often to an 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 3.20, we find K_{10} byte values. The equation 3.20 is calculated from the first correct ciphertext equation 3.10.

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

Now, we examine the practical experiment for this DFA.

3.6.2 Practical Experiment

For this experiment, it is required to perform these three steps:

1. **Correct encryption:** It is necessary to obtain the correct ciphertexts for at least 3 plaintexts.

2. **Fault injection on K_9:** It is required to 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 protect physically the previous round keys from fault injection, but there is not any restriction for K_9 and K_{10}. The the laser emission can be done at any time in t_9 or before. The figure 3.20 shows this attack's exploration.

3. **Faulty encryption:** It is necessary to encipher again the same plaintexts and obtain their faulty ciphertexts without circuit reset.

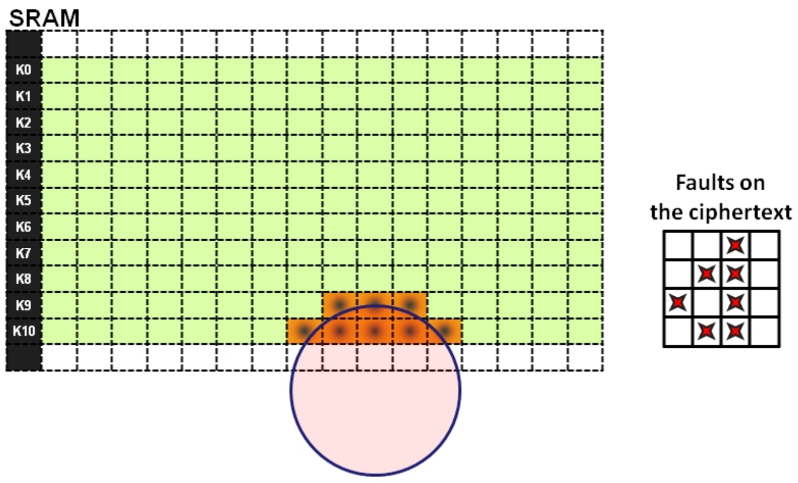


Figure 3.20: Attack's exploration for this extended multi-byte DFA. The laser emission time is set to t_9 during exploration.

Therefore, each faulty byte of K_9 reveals the content of its respective K_{10} byte after 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. But, 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 the 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 protect physically all the round keys prior to K_9 from fault injection. Nevertheless, this attack is still feasible but with longer practical experiments.

In this case, it is required to perform these different stages:

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. **Fault injection on K_9 and creating the list of faulty ciphertexts:** It is required to launch a series of tests by fault injection on K_9 bytes and performing two encryptions per 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. But, the second faulty ciphertext includes all the faults on the round keys.

   For the first encryptions, the opponent uses repetitively 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 again repeating the same sequence: M^a for the fourth test, etc.

   For the second encryptions, the opponent uses repetitively the first plaintext; *i.e.* M^a, as the reference text. Therefore, if at two 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 products of same faults on K_{10} and K_9. The second encryptions are also products of same faults on K_{10}, K_9 and prior round keys.

   Thus, the opponent creates a list of three values: index of the first plaintext, first faulty ciphertext and 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 plaintext at first encryptions, the experiments 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.

The figure 3.21 shows this attack's exploration. The part of this 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.

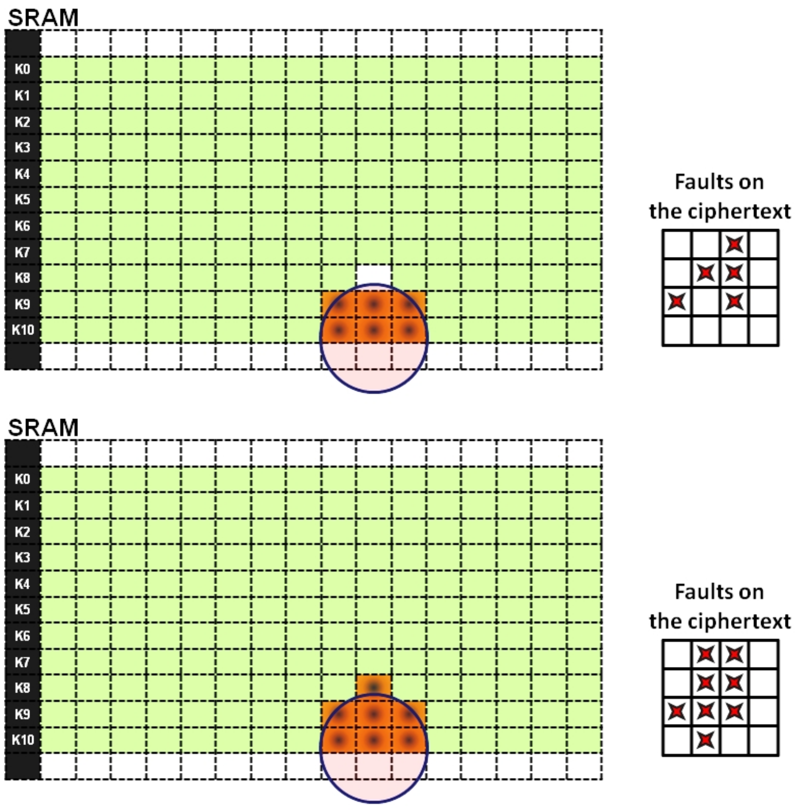


Figure 3.21: Attack's exploration for the more sophisticated case of our extended multi-byte DFA. The laser emission time is set to t_9 for the first encryption during exploration. The first and the second encryptions are shown in top and bottom part of figure respectively.

At the first look, this experiment seems to be very time consuming. But, in practice for our circuit, it 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 high chance of reproducing similar faults 3 times in a short time.

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

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

Therefore, if each combination has a chance equal to other ones, in the worse case, after 1512 $\times$ 2 + 1 = 3024 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 always feasible by tricks in 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 Conclusions

In this chapter, we surveyed about the feasibility of Piret-Quisquater and Giraud's bit methods that are single-byte and single-bit DFA respectively, by laser beam that hits several bytes.

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 targeted round key and also on the previous and the following ones in comparison to the required model.

We excluded logically the effect of any fault occurred on the previous round keys by a laser emission time tuning. 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 μm precision for displacements.

The main conclusion until this part of our experiment was that by a big laser spot that hits several bytes, it is still possible to perform single-byte or even single-bit DFA. Instead of investing for an accurate laser facility, an opponent may use a cheap laser emitter and compensate its lack of precision by investing on 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 help again to perform successfully DFA on AES.

The laser spot diameter and fabrication technology size are important parameters in 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 tricks on unprotected chips. They might be done even by big laser spots. Therefore, developing the proper countermeasures against laser fault attacks is necessary for the security of cryptographic circuits.

Chapter 4

Round Modification Attacks

Contents

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. There are very few other works that followed this attack, the most notable 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 using faults by a laser bench. The experiments are done on our circuit running SOSSE operating system with an embedded AES.

4.1.1 Previous Works

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 may change the round counter (therefore RC) of AES at the beginning of algorithm execution to its final value (0 in the [Choukri 2005] case), the ciphertext will be the product of a single round (after the initial round).

Regarding our AES implementation, one of two different algorithm executions may happen:

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:

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

   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:

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

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 beginning of the AES execution.

4.1.1.2 Attack of Y. Monnet et al.

Y. Monnet et al. are reported in [Monnet 2006] another round reduction attack on two asynchronous cryptoprocessors running the DES algorithm. The round counters used in both of cryptoprocessors have a multi-rail design. For each of 16 rounds, a separated signal line is implemented. A 17th signal line is also existent to distinguish the end of algorithm execution. These signal lines are protected by an alarm that releases 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 against attacks during the experiments. However, this attack is done successfully on both of the circuits.

4.1.1.3 Attack of J. H. Park et al.

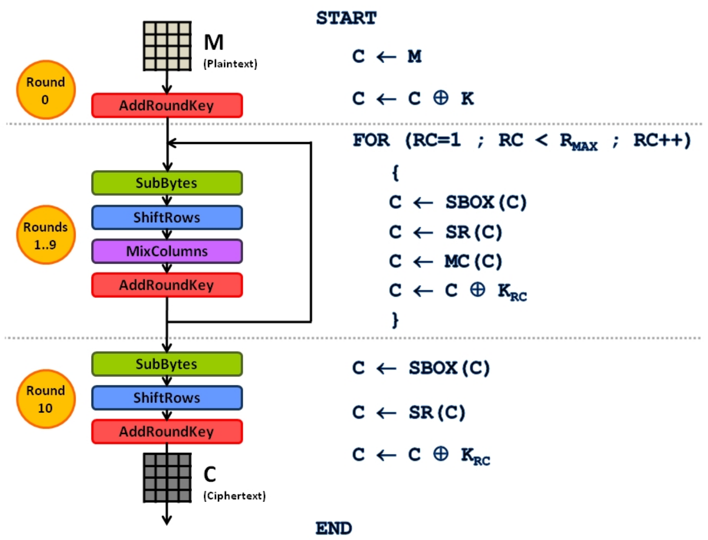
The attack of JeaHoon Park et al. reported in [Park 2011] is a laser fault attack on an embedded AES using an Atmega128 microcontroller. The AES implementation is based on the proposed one in the NIST reference [NIST 2001].

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 attaque 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 on the figure 4.1:



```
START
    C ← M
    C ← C ⊕ K
    FOR (RC=1 ; RC < R_MAX ; RC++)
        {
        C ← SBOX(C)
        C ← SR(C)
        C ← MC(C)
        C ← C ⊕ K_RC
        }
    C ← SBOX(C)
    C ← SR(C)
    C ← C ⊕ K_RC
END
```

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 the figure 4.1. Therefore, even with complete removal of middle rounds, the initial and the final rounds will be executed.

However, the index of 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 for 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 the set of these hexadecimal values: {`01, 02, 04, 08, 10, 20, 40, 80`} and omit other values with fewer feasibility.

The fault injection moment has distinct effects regarding 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, between RC value assignment or incrementation and its comparison to the upper limit, so called "R_{max}" value.
2. LOOP : After RC comparison to its upper limit and before `AddRoundKey` operation of the current round.
3. ARK : During the `AddRoundKey` operation for the current round.
4. COMEBACK : After the `AddRoundKey` operation, until the return to the loop's beginning.

We show this various moments of our AES algorithm execution on the figure 4.2.

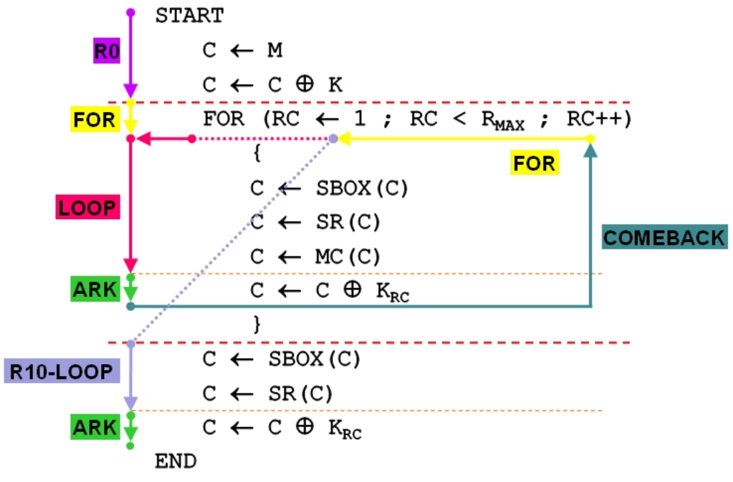


Figure 4.2: Various moments of our AES algorithm execution.

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 these so called *single attack scenarios* as they requires 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 on the algorithm execution. Changing the RC value often leads to change of total number of executed rounds, by adding, suppressing or even repetitive executing of 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:

  $$\mathrm{R}_0..\mathrm{R}_5\text{-}\mathrm{R}_6\text{-}\mathrm{R}_5\text{-}\mathrm{R}_6\text{-}\mathrm{R}_7..\mathrm{R}_{10}$$

  The rounds 5 and 6 will be executed twice and the total number of executed rounds will be incremented 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:

  $$\mathrm{R}_0..\mathrm{R}_3\text{-}\mathrm{R}_6..\mathrm{R}_{10}$$

  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.* without any change on total number of rounds, but with effects on `AddRoundKey` of final round and maybe penultimate round. For instance: if RC=9 and e=2 then the new RC=11 and the AES execution will be:

  $$\mathrm{R}_0..\mathrm{R}_8\text{-}\mathrm{R}_{m=11}\text{-}\mathrm{R}_{f=12}$$

  So, the total number of executed rounds will remain 10, but the penultimate round and the final round will xored by invalid subkey values 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.

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

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

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)

- **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 it is 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 it is 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 it is 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 it is 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 refer to attack moment and value of the injected fault.

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

The second line present list of executed rounds. For brevity, consecutive normal rounds are replaced by two points in the lists, 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 similarly to final round is executed. The next one or two lines show execution condition of faulty rounds. For the current example, the next line refers to the faulty final round; *e.g.* when f=25, the `AddRoundKey` operation of final round searches for subkey values corresponding to round 25.

Any round does not exist after the round 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 corresponding bloc of 16 bytes on memory for any subkey index.

As another example, R_0-R_1-R_m-R_f means that totally 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 the rounds 17 and 18.

Table 4.2 shows a brief comparison between the correct and a faulty round executions.

Table 4.2: Comparison between the correct and a faulty round executions.

The correct execution	Example for a round modified execution

NR=10
$R_0..R_{10}$
	$NR \to 3$
	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
$RC = 1$		$RC\to 0$	$RC\to 3$	$RC\to 5$	$RC\to 9$	$RC\to 17$	$RC\to 33$	$RC\to 65$	$RC\to 129$
	FOR					$NR\to 1$ R_0-R_f f=17	$NR\to 1$ R_0-R_f f=33	$NR\to 1$ R_0-R_f f=65	$NR\to 1$ R_0-R_f f=129
	LOOP					$NR\to 2$ R_0-R_m-R_f m=17 f=18	$NR\to 2$ R_0-R_m-R_f m=33 f=34	$NR\to 2$ R_0-R_m-R_f m=65 f=66	$NR\to 2$ R_0-R_m-R_f m=129 f=130
	ARK COMEBACK					$NR\to 2$ R_0-R_1-R_f f=18	$NR\to 2$ R_0-R_1-R_f f=34	$NR\to 2$ R_0-R_1-R_f f=66	$NR\to 2$ R_0-R_1-R_f f=130
$RC = 2$		$RC\to 3$	$RC\to 0$	$RC\to 6$	$RC\to 10$	$RC\to 18$	$RC\to 34$	$RC\to 66$	$RC\to 130$
	FOR					$NR\to 2$ R_0-R_1-R_f f=18	$NR\to 2$ R_0-R_1-R_f f=34	$NR\to 2$ R_0-R_1-R_f f=66	$NR\to 2$ R_0-R_1-R_f f=130
	LOOP					$NR\to 3$ R_0-R_1-R_m-R_f m=18 f=19	$NR\to 3$ R_0-R_1-R_m-R_f m=34 f=35	$NR\to 3$ R_0-R_1-R_m-R_f m=66 f=67	$NR\to 3$ R_0-R_1-R_m-R_f m=130 f=131

Table continued on next page

Table 4.3 – Table continued from previous page

Round counter	Attack moment	Model of injected fault							
		0x01	0x02	0x04	0x08	0x10	0x20	0x40	0x80
$RC = 8$		$RC\to 9$	$RC\to 10$	$RC\to 12$	$RC\to 0$	$RC\to 24$	$RC\to 40$	$RC\to 72$	$RC\to 136$
	ARK COMEBACK		$NR\to 9$ $R_0..R_8$-R_f f=11	$NR\to 9$ $R_0..R_8$-R_f f=13		$NR\to 9$ $R_0..R_8$-R_f f=25	$NR\to 9$ $R_0..R_8$-R_f f=41	$NR\to 9$ $R_0..R_8$-R_f f=73	$NR\to 9$ $R_0..R_8$-R_f f=137
$RC = 9$		$RC\to 8$	$RC\to 11$	$RC\to 13$	$RC\to 1$	$RC\to 25$	$RC\to 41$	$RC\to 73$	$RC\to 137$
	FOR		$NR\to 9$ $R_0..R_8$-R_f f=11	$NR\to 9$ $R_0..R_8$-R_f f=13		$NR\to 9$ $R_0..R_8$-R_f f=25	$NR\to 9$ $R_0..R_8$-R_f f=41	$NR\to 9$ $R_0..R_8$-R_f f=73	$NR\to 9$ $R_0..R_8$-R_f f=137
	LOOP		$NR\to 10$ $R_0..R_8$-R_m-R_f m=11 f=12	$NR\to 10$ $R_0..R_8$-R_m-R_f m=13 f=14		$NR\to 10$ $R_0..R_8$-R_m-R_f m=25 f=26	$NR\to 10$ $R_0..R_8$-R_m-R_f m=41 f=42	$NR\to 10$ $R_0..R_8$-R_m-R_f m=73 f=74	$NR\to 10$ $R_0..R_8$-R_m-R_f m=137 f=138
	ARK COMEBACK		$NR\to 10$ $R_0..R_9$-R_f f=12	$NR\to 10$ $R_0..R_9$-R_f f=14		$NR\to 10$ $R_0..R_9$-R_f f=26	$NR\to 10$ $R_0..R_9$-R_f f=42	$NR\to 10$ $R_0..R_9$-R_f f=74	$NR\to 10$ $R_0..R_9$-R_f f=138
$RC = 10$		$RC\to 11$	$RC\to 8$	$RC\to 14$	$RC\to 2$	$RC\to 26$	$RC\to 42$	$RC\to 74$	$RC\to 138$
	FOR	$NR\to 10$ $R_0..R_9$-R_f f=11		$NR\to 10$ $R_0..R_9$-R_f f=14		$NR\to 10$ $R_0..R_9$-R_f f=26	$NR\to 10$ $R_0..R_9$-R_f f=42	$NR\to 10$ $R_0..R_9$-R_f f=74	$NR\to 10$ $R_0..R_9$-R_f f=138
	LOOP	$NR\to 10$ $R_0..R_9$-R_f f=11		$NR\to 10$ $R_0..R_9$-R_f f=14		$NR\to 10$ $R_0..R_9$-R_f f=26	$NR\to 10$ $R_0..R_9$-R_f f=42	$NR\to 10$ $R_0..R_9$-R_f f=74	$NR\to 10$ $R_0..R_9$-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, it can never prevent the execution of final round. However, depending to 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)

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

The consequences of injecting different values of faults on the round number reference are shown in the Annexe 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 LOOP ARK COMEBACK	$NR\rightarrow 11$ $R_0..R_9$-R_m-R_f m=10 f=11		$NR\rightarrow 14$ $R_0..R_9$-$4\times R_m$-R_f m={10, 11, 12, 13} f=14	
$RC = 8$					
	FOR	$NR\rightarrow 11$ $R_0..R_9$-R_m-R_f m=10 f=11		$NR\rightarrow 14$ $R_0..R_9$-$4\times R_m$-R_f m={10, 11, 12, 13} 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
		$R_{max}\to 11$	$R_{max}\to 8$	$R_{max}\to 14$	$R_{max}\to 2$
	LOOP ARK COMEBACK	$NR\to 11$ $R_0..R_9-R_m-R_f$ m=10 f=11	$NR\to 9$ $R_0..R_8-R_f$ f=9	$NR\to 14$ $R_0..R_9-4\times R_m-R_f$ m={10, 11, 12, 13} f=14	
$RC = 9$					
	FOR	$NR\to 11$ $R_0..R_9-R_m-R_f$ m=10 f=11	$NR\to 9$ $R_0..R_8-R_f$ f=9	$NR\to 14$ $R_0..R_9-4\times R_m-R_f$ m={10, 11, 12, 13} f=14	
	LOOP ARK COMEBACK	$NR\to 11$ $R_0..R_9-R_m-R_f$ m=10 f=11		$NR\to 14$ $R_0..R_9-4\times R_m-R_f$ m={10, 11, 12, 13} f=14	
$RC = 10$					
	FOR	$NR\to 11$ $R_0..R_9-R_m-R_f$ m=10 f=11		$NR\to 14$ $R_0..R_9-4\times R_m-R_f$ 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. But we can 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 to the memory and overwrite zero (`0x00`) or any known values on the addresses upper the stored round keys (at least for the corresponding bytes to the attack).

Or, if the memory content for the required non valid round key is equal to zero or any known value.

A third similar possibility is when the corresponding address for a non-valid round key is out of the memory space. For instance, if according to the attack, the algorithm searches for K_{129} and its corresponding address is out of the memory space. In this case, if 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. We describe here 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:

$$D = \texttt{SR} \circ \texttt{SB}(M \oplus K) \tag{4.1}$$

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

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

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

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

D and M have known values.

Therefore, each byte of K is revealed by the 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:

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

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

Each faulty ciphertext has an xor operation with K'_y, 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:

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

Then, by reversing `ShiftRows` operation in equation 4.5 and according to its distributivity property, we obtain equation 4.6:

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

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 give 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 only one corresponding key.

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

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

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

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:

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

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

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

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

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

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

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

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 faulty ciphertext and its corresponding plaintext are required:

$$D^a = \mathtt{SR} \circ \mathtt{SB}\{\mathtt{MC} \circ \mathtt{SR} \circ \mathtt{SB}(M^a \oplus K) \oplus K'_x\} \oplus K'_y \tag{4.13a}$$
$$D^b = \mathtt{SR} \circ \mathtt{SB}\{\mathtt{MC} \circ \mathtt{SR} \circ \mathtt{SB}(M^b \oplus K) \oplus K'_x\} \oplus K'_y \tag{4.13b}$$
$$D^c = \mathtt{SR} \circ \mathtt{SB}\{\mathtt{MC} \circ \mathtt{SR} \circ \mathtt{SB}(M^c \oplus K) \oplus K'_x\} \oplus K'_y \tag{4.13c}$$

Each faulty ciphertext, shown as an equation 4.13, has two invalid key values. This attacks results in 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:

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

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

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:

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

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

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

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

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

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

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

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 that only one key byte value is correct, both of these two byte values satisfy the 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 examinated using one of the equations 4.15. Therefore, K_1 expansion is needed for 2^{16} entire key hypotheses in order to find the only one 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 the equation 4.19. So, the brute-force search for the key byte values on both of equations 4.24 and 4.20 gives only the correct key byte value.

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

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

- **Attack I-3 without memory access**

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

The equations 4.15 show two first faulty ciphertexts. Each faulty ciphertext has K'_y, as a invalid key value:

$$D^a = \texttt{SR}\circ\texttt{SB}[\texttt{MC}\circ\texttt{SR}\circ\texttt{SB}(M^a\oplus K)\oplus K_1]\oplus K'_y\tag{4.21a}$$
$$D^b = \texttt{SR}\circ\texttt{SB}[\texttt{MC}\circ\texttt{SR}\circ\texttt{SB}(M^b\oplus K)\oplus K_1]\oplus K'_y\tag{4.21b}$$
$$D^c = \texttt{SR}\circ\texttt{SB}[\texttt{MC}\circ\texttt{SR}\circ\texttt{SB}(M^c\oplus K)\oplus K_1]\oplus K'_y\tag{4.21c}$$

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

$$D^a \oplus K'_y = \texttt{SR} \circ \texttt{SB}[\texttt{MC} \circ \texttt{SR} \circ \texttt{SB}(M^a \oplus K) \oplus K_1] \tag{4.22a}$$
$$D^b \oplus K'_y = \texttt{SR} \circ \texttt{SB}[\texttt{MC} \circ \texttt{SR} \circ \texttt{SB}(M^b \oplus K) \oplus K_1] \tag{4.22b}$$
$$D^c \oplus K'_y = \texttt{SR} \circ \texttt{SB}[\texttt{MC} \circ \texttt{SR} \circ \texttt{SB}(M^c \oplus K) \oplus K_1] \tag{4.22c}$$

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

$$\texttt{SB}^{-1} \circ \texttt{SR}^{-1}(D^a \oplus K'_y) = \texttt{MC} \circ \texttt{SR} \circ \texttt{SB}(M^a \oplus K) \oplus K_1 \tag{4.23a}$$
$$\texttt{SB}^{-1} \circ \texttt{SR}^{-1}(D^b \oplus K'_y) = \texttt{MC} \circ \texttt{SR} \circ \texttt{SB}(M^b \oplus K) \oplus K_1 \tag{4.23b}$$
$$\texttt{SB}^{-1} \circ \texttt{SR}^{-1}(D^c \oplus K'_y) = \texttt{MC} \circ \texttt{SR} \circ \texttt{SB}(M^c \oplus K) \oplus K_1 \tag{4.23c}$$

We can discard K_1 value from equations 4.23 by an xor operation between them and arrive to the equation 4.24:

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

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

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

We create hypotheses by a brute-force search on each quadruple of K'_y bytes that enters together to a single `MixColumns` operation. Then, for each hypothesis on a K'_y quadruple, we search values for the 4 corresponding bytes of K. As there is not any `MixColumns` step for K values in the equation 4.24, we can perform our brute-force search for each of bytes on the quadruple separately.

So, for finding each K'_y quadruple, $(2^8)^4$ values must be verified. In addition, finding each byte of a K quadruple requires a brute-force search between 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:

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

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

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

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

$$\mathtt{SB}^{-1} \circ \mathtt{SR}^{-1} \circ \mathtt{MC}^{-1} \circ \mathtt{SB}^{-1} \circ \mathtt{SR}^{-1}(D^a) = [\mathtt{MC} \circ \mathtt{SR} \circ \mathtt{SB}(M^a \oplus K)] \oplus K_1 \tag{4.27a}$$
$$\mathtt{SB}^{-1} \circ \mathtt{SR}^{-1} \circ \mathtt{MC}^{-1} \circ \mathtt{SB}^{-1} \circ \mathtt{SR}^{-1}(D^b) = [\mathtt{MC} \circ \mathtt{SR} \circ \mathtt{SB}(M^b \oplus K)] \oplus K_1 \tag{4.27b}$$

Afterwards, we can discard the effect of K_1 in the equations 4.27 by an xor operation between them and obtain the equation 4.28:

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

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

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

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

So, for each column of 4 bytes on K in the equation 4.29, a brute-force search between $(2^8)^4$ 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 the equations 4.30, has K'_x and K'_y, two invalid key values.

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

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

$$\begin{aligned} D^a \oplus D^b = \\ \texttt{SR} \circ \texttt{SB}\{\texttt{MC} \circ \texttt{SR} \circ \texttt{SB}[\texttt{MC} \circ \texttt{SR} \circ \texttt{SB}(M^a \oplus K) \oplus K_1] \oplus K_x\} \oplus \\ \texttt{SR} \circ \texttt{SB}\{\texttt{MC} \circ \texttt{SR} \circ \texttt{SB}[\texttt{MC} \circ \texttt{SR} \circ \texttt{SB}(M^b \oplus K) \oplus K_1] \oplus K_x\} \end{aligned} \tag{4.31}$$

K'_x cannot be discarded from the equation 4.31. Searching for both of K and K_1 in presence of the invalid key values of K'_x and two `MixColumns` steps cannot be limited to a partial 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:

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

$$D^a = \texttt{SR} \circ \texttt{SB}(M_8^a) \oplus K_{10} \tag{4.33a}$$
$$D^b = \texttt{SR} \circ \texttt{SB}(M_8^b) \oplus K_{10} \tag{4.33b}$$

According to substitution property of xor, we can move K_{10} in the equations4.32 and obtain the equations 4.34:

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

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

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

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

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

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

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

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 the equations 4.38 and 4.39.

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

$$D^c = \mathtt{SR} \circ \mathtt{SB}(M_8^c) \oplus K_{10} \tag{4.39a}$$

Therefore, using both of the equations 4.37 and 4.40 leads to a unique value for each of K_{10} byte (with some exceptions).

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

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:

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

$$D^a = \mathtt{SR} \circ \mathtt{SB}(M_8^a) \oplus 00 \tag{4.42a}$$
$$D^b = \mathtt{SR} \circ \mathtt{SB}(M_8^b) \oplus 00 \tag{4.42b}$$

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

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

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

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

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 the equation 4.44 gives two hypotheses on each byte of K_9. In this case, 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, solving both of equations 4.44 and 4.47 leads to a single value for each byte of K_9.

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

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

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

However, in some exceptional cases, even by using 3 pairs of corresponding faulty and correct ciphertexts, more than one byte might 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:

$$C^a = \mathtt{SR} \circ \mathtt{SB}[\mathtt{MC} \circ \mathtt{SR} \circ \mathtt{SB}(M_8^a) \oplus K_9] \oplus K_{10} \tag{4.48a}$$
$$C^b = \mathtt{SR} \circ \mathtt{SB}[\mathtt{MC} \circ \mathtt{SR} \circ \mathtt{SB}(M_8^b) \oplus K_9] \oplus K_{10} \tag{4.48b}$$
$$C^c = \mathtt{SR} \circ \mathtt{SB}[\mathtt{MC} \circ \mathtt{SR} \circ \mathtt{SB}(M_8^c) \oplus K_9] \oplus K_{10} \tag{4.48c}$$

$$D^a = \mathtt{SR} \circ \mathtt{SB}(M_8^a) \oplus K'_y \tag{4.49a}$$
$$D^b = \mathtt{SR} \circ \mathtt{SB}(M_8^b) \oplus K'_y \tag{4.49b}$$
$$D^c = \mathtt{SR} \circ \mathtt{SB}(M_8^c) \oplus K'_y \tag{4.49c}$$

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

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

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

$$C^a \oplus C^b = \texttt{SR} \circ \texttt{SB}[\texttt{MC} \circ \texttt{SR} \circ \texttt{SB}(M_8^a) \oplus K_9] \oplus \\ \texttt{SR} \circ \texttt{SB}[\texttt{MC} \circ \texttt{SR} \circ \texttt{SB}(M_8^b) \oplus K_9] \tag{4.51a}$$

$$C^a \oplus C^c = \texttt{SR} \circ \texttt{SB}[\texttt{MC} \circ \texttt{SR} \circ \texttt{SB}(M_8^a) \oplus K_9] \oplus \\ \texttt{SR} \circ \texttt{SB}[\texttt{MC} \circ \texttt{SR} \circ \texttt{SB}(M_8^c) \oplus K_9] \tag{4.51b}$$

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

$$X^a = \texttt{SR} \circ \texttt{SB}(M_8^a) \tag{4.52a}$$
$$X^b = \texttt{SR} \circ \texttt{SB}(M_8^b) \tag{4.52b}$$
$$X^c = \texttt{SR} \circ \texttt{SB}(M_8^c) \tag{4.52c}$$

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:

$$D^a \oplus D^b = X^a \oplus X^b \tag{4.53a}$$
$$D^a \oplus D^c = X^a \oplus X^c \tag{4.53b}$$

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

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

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

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:

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

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

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

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

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 the new equations 4.58:

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

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

C^a, C^b, C^c, D^a, D^b and D^c are 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 gives often one value for each byte of K_{10}. Nevertheless, in exceptional cases, it might give two hypotheses for several bytes of K_{10}. The non-corresponding hypotheses can be identified and discarded by a new exhaustive search between 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:

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

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

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

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

An xor operation between each two correct ciphertexts 4.59, excludes K_{10} and leads to equation 4.61:

$$C^a \oplus C^b = \mathtt{SR} \circ \mathtt{SB}[\mathtt{MC} \circ \mathtt{SR} \circ \mathtt{SB}(M_8^a) \oplus K_9] \oplus \\ \mathtt{SR} \circ \mathtt{SB}[\mathtt{MC} \circ \mathtt{SR} \circ \mathtt{SB}(M_8^b) \oplus K_9] \tag{4.61}$$

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

$$\mathtt{SB}^{-1} \circ \mathtt{SR}^{-1}(D^a) = \mathtt{MC} \circ \mathtt{SR} \circ \mathtt{SB}(M_8^a) \tag{4.62a}$$
$$\mathtt{SB}^{-1} \circ \mathtt{SR}^{-1}(D^b) = \mathtt{MC} \circ \mathtt{SR} \circ \mathtt{SB}(M_8^b) \tag{4.62b}$$

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

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

C^a, C^b, D^a and D^b are 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 gives often 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 only between 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:

$$C^a = \mathtt{SR} \circ \mathtt{SB}(M_9^a) \oplus K_{10} = \mathtt{SR} \circ \mathtt{SB}[\mathtt{MC} \circ \mathtt{SR} \circ \mathtt{SB}(M_8^a) \oplus K_9] \oplus K_{10} \tag{4.64a}$$
$$C^b = \mathtt{SR} \circ \mathtt{SB}(M_9^b) \oplus K_{10} = \mathtt{SR} \circ \mathtt{SB}[\mathtt{MC} \circ \mathtt{SR} \circ \mathtt{SB}(M_8^b) \oplus K_9] \oplus K_{10} \tag{4.64b}$$
$$C^c = \mathtt{SR} \circ \mathtt{SB}(M_9^c) \oplus K_{10} = \mathtt{SR} \circ \mathtt{SB}[\mathtt{MC} \circ \mathtt{SR} \circ \mathtt{SB}(M_8^c) \oplus K_9] \oplus K_{10} \tag{4.64c}$$

$$D^a = \mathtt{SR} \circ \mathtt{SB}[\mathtt{MC} \circ \mathtt{SR} \circ \mathtt{SB}(M_8^a) \oplus K'_x] \oplus K'_y \tag{4.65a}$$
$$D^b = \mathtt{SR} \circ \mathtt{SB}[\mathtt{MC} \circ \mathtt{SR} \circ \mathtt{SB}(M_8^b) \oplus K'_x] \oplus K'_y \tag{4.65b}$$
$$D^c = \mathtt{SR} \circ \mathtt{SB}[\mathtt{MC} \circ \mathtt{SR} \circ \mathtt{SB}(M_8^c) \oplus K'_x] \oplus K'_y \tag{4.65c}$$

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

$$K'_x = K_9 \oplus E_9 \tag{4.66}$$

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

$$D^a = \texttt{SR} \circ \texttt{SB}[(M_9^a) \oplus E_9] \oplus K'_y \tag{4.67a}$$
$$D^b = \texttt{SR} \circ \texttt{SB}[(M_9^b) \oplus E_9] \oplus K'_y \tag{4.67b}$$
$$D^c = \texttt{SR} \circ \texttt{SB}[(M_9^c) \oplus E_9] \oplus K'_y \tag{4.67c}$$

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

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

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

$$C^a \oplus C^b = \texttt{SR} \circ \texttt{SB}(M_9^a) \oplus \texttt{SR} \circ \texttt{SB}(M_9^b) \tag{4.69a}$$
$$C^a \oplus C^c = \texttt{SR} \circ \texttt{SB}(M_9^a) \oplus \texttt{SR} \circ \texttt{SB}(M_9^c) \tag{4.69b}$$

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. So, we obtain equations 4.70:

$$M_9^b = \texttt{SB}^{-1}[\texttt{SR}^{-1}(C^a \oplus C^b) \oplus \texttt{SB}(M_9^a)] \tag{4.70a}$$
$$M_9^c = \texttt{SB}^{-1}[\texttt{SR}^{-1}(C^a \oplus C^c) \oplus \texttt{SB}(M_9^a)] \tag{4.70b}$$

We replace obtained for M_9^b and M_9^c by equations 4.70 in equations 4.68 and so we get equations 4.71:

$$\begin{gathered}\texttt{SR}^{-1}(D^a \oplus D^b) = \texttt{SB}[(M_9^a) \oplus E_9] \oplus \\ \texttt{SB}[\texttt{SB}^{-1}(\texttt{SR}^{-1}(C^a \oplus C^b) \oplus \texttt{SB}(M_9^a)) \oplus E_9]\end{gathered} \tag{4.71a}$$

$$\begin{aligned}\texttt{SR}^{-1}(D^a \oplus D^c) = \texttt{SB}[(M_9^a) \oplus E_9]\oplus \\ \texttt{SB}[\texttt{SB}^{-1}(\texttt{SR}^{-1}(C^a \oplus C^c) \oplus \texttt{SB}(M_9^a)) \oplus E_9]\end{aligned} \tag{4.71b}$$

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

Finally, we perform an exhaustive search between 2^8 possible values for each M_9^a byte and between 2^8 possible values for each corresponding E_9 byte. This exhaustive search leads often to an 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.

$$K_{10} = \texttt{SR} \circ \texttt{SB}(M_9^a) \oplus C^a \tag{4.72}$$

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:

$$C = \texttt{SR} \circ \texttt{SB}(M^9) \oplus K_{10} \tag{4.73}$$

$$D = \texttt{SR} \circ \texttt{SB}(M^9) \oplus 00 \tag{4.74}$$

We can obtain K_{10} value by performing only an xor operation between equation 4.73 and equation 4.74:

$$K_{10} = C \oplus D \tag{4.75}$$

C and D are 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 the equations 4.76 and 4.77 respectively. As K'_y value is unknown and cannot be revealed by any operation between correct and faulty ciphertexts. So, there is not any possibility fo find K_{10} or other key values in this attack and it is not exploitable without memory access.

$$C = \texttt{SR} \circ \texttt{SB}(M^9) \oplus K_{10} \tag{4.76}$$

$$D = \texttt{SR} \circ \texttt{SB}(M^9) \oplus K'_y \tag{4.77}$$

4.2.2.2 Attacks on The Round Number Reference

For the second scenario, again, both of attack possibilities with or without memory access can be considered. However, similarly to the first scenario, independent attacks are also existent.

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 more easiness, we assume `MiddleRound` and `FinalRound` as two functions that combine different operations of a middle of a final round. We present them as equations 4.78 and 4.79:

$$\texttt{MiddleRound}(M) \text{ or } \texttt{MR}(M) = \texttt{MC} \circ \texttt{SR} \circ \texttt{SB}(M) \tag{4.78}$$

$$\texttt{FinalRound}(M) \text{ or } \texttt{FR}(M) = \texttt{SR} \circ \texttt{SB}(M) \tag{4.79}$$

The reversed functions of `MiddleRound` and `FinalRound` also represent the reverse of round operations, as shown in equations 4.80 and 4.81:

$$\texttt{MiddleRound}^{-1}(D) \text{ or } \texttt{MR}^{-1}(D) = \texttt{SB}^{-1} \circ \texttt{SR}^{-1} \circ \texttt{MC}^{-1}(D) \tag{4.80}$$

$$\texttt{FinalRound}^{-1}(C) \text{ or } \texttt{FR}^{-1}(C) = \texttt{SB}^{-1} \circ \texttt{SR}^{-1}(C) \tag{4.81}$$

We will use several times `MiddleRound` and `FinalRound` 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: $R_0..R_9\text{-}R_{m=10}\text{-}R_{f=11}$

- **Attack II-1 with memory access**

For exploiting this attack, only one pair of corresponding faulty and correct ciphertexts is required:

$$C = \texttt{SR} \circ \texttt{SB}(M_9) \oplus K_{10} \tag{4.82}$$

$$D = \texttt{SR} \circ \texttt{SB}[\texttt{MC} \circ \texttt{SR} \circ \texttt{SB}(M_9) \oplus K_{10}] \oplus 00 \tag{4.83}$$

By reversing `ShiftRows` and `SubBytes` operations in equation 4.83, we get the equation 4.84:

$$\mathtt{SB}^{-1} \circ \mathtt{SR}^{-1}(D) = \mathtt{MC} \circ \mathtt{SR} \circ \mathtt{SB}(M_9) \oplus K_{10} \tag{4.84}$$

Then, an xor operation between equation 4.84 and equation 4.82 give the new equation 4.85:

$$C \oplus \mathtt{SB}^{-1} \circ \mathtt{SR}^{-1}(D) = \mathtt{SR} \circ \mathtt{SB}(M_9) \oplus \mathtt{MC} \circ \mathtt{SR} \circ \mathtt{SB}(M_9) \tag{4.85}$$

We define X value by equation 4.86 for more easiness in writing the equations:

$$X = \mathtt{SR} \circ \mathtt{SB}(M_9) \tag{4.86}$$

We replace X in equation 4.85 and get the new equation 4.87:

$$C \oplus \mathtt{SB}^{-1} \circ \mathtt{SR}^{-1}(D) = X \oplus \mathtt{MC}(X) \tag{4.87}$$

Then, we perform an exhaustive search on values of X in equation 4.87. As there is a `MixColumns` step in equation 4.87, the value of each byte of X depends to the other values of the same column. So, we must perform the exhaustive search for the quadruples of 4 bytes of the first column. Then, we repeat it for each of next three columns.

Each column hypothesis on X gives an hypothesis on the corresponding column of K_{10} by equation 4.88. This exhaustive search between $(2^8)^4$ values on each column of X leads to 2^8 hypotheses on it and consequently on the corresponding column of K_{10}.

$$K_{10} = C \oplus X \tag{4.88}$$

At the next step, a second exhaustive search is required between all the combinations of column hypotheses on K_{10}. To perform this new search, for each combination of 4 columns hypothesis on 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 interrupts.

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 at the second step. However, in average the key is revealed after $(2^8)^4$ searches. So, a first exhaustive search of 2^{34} values continued by a second search of 2^{31} in average is required for finding the key.

- **Attack II-1 without memory access**

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

$$C^a = \texttt{SR} \circ \texttt{SB}(M_9^a) \oplus K_{10} \tag{4.89a}$$
$$C^b = \texttt{SR} \circ \texttt{SB}(M_9^b) \oplus K_{10} \tag{4.89b}$$
$$C^c = \texttt{SR} \circ \texttt{SB}(M_9^c) \oplus K_{10} \tag{4.89c}$$

$$D^a = \texttt{SR} \circ \texttt{SB}[\texttt{MC} \circ \texttt{SR} \circ \texttt{SB}(M_9^a) \oplus K_{10}] \oplus K'_{f=11} \tag{4.90a}$$
$$D^b = \texttt{SR} \circ \texttt{SB}[\texttt{MC} \circ \texttt{SR} \circ \texttt{SB}(M_9^b) \oplus K_{10}] \oplus K'_{f=11} \tag{4.90b}$$
$$D^c = \texttt{SR} \circ \texttt{SB}[\texttt{MC} \circ \texttt{SR} \circ \texttt{SB}(M_9^c) \oplus K_{10}] \oplus K'_{f=11} \tag{4.90c}$$

Xor operations between the first faulty ciphertext equation and each of next ones in equations 4.90 give the new equations 4.91:

$$\begin{aligned} D^a \oplus D^b = {} & \texttt{SR} \circ \texttt{SB}[\texttt{MC} \circ \texttt{SR} \circ \texttt{SB}(M_9^a) \oplus K_{10}] \oplus \\ & \texttt{SR} \circ \texttt{SB}[\texttt{MC} \circ \texttt{SR} \circ \texttt{SB}(M_9^b) \oplus K_{10}] \end{aligned} \tag{4.91a}$$

$$\begin{aligned} D^a \oplus D^c = {} & \texttt{SR} \circ \texttt{SB}[\texttt{MC} \circ \texttt{SR} \circ \texttt{SB}(M_9^a) \oplus K_{10}] \oplus \\ & \texttt{SR} \circ \texttt{SB}[\texttt{MC} \circ \texttt{SR} \circ \texttt{SB}(M_9^c) \oplus K_{10}] \end{aligned} \tag{4.91b}$$

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:

$$\texttt{SR}^{-1}(D^a \oplus D^b) = \texttt{SB}[\texttt{MC}(C^a \oplus K_{10}) \oplus K_{10}] \oplus \texttt{SB}[\texttt{MC}(C^b \oplus K_{10}) \oplus K_{10}] \tag{4.92a}$$
$$\texttt{SR}^{-1}(D^a \oplus D^c) = \texttt{SB}[\texttt{MC}(C^a \oplus K_{10}) \oplus K_{10}] \oplus \texttt{SB}[\texttt{MC}(C^c \oplus K_{10}) \oplus K_{10}] \tag{4.92b}$$

According to the distributivity property of `MixColumns`, we transform equations 4.85 to the new equations 4.93:

$$\begin{aligned} \texttt{SR}^{-1}(D^a \oplus D^b) = {} & \texttt{SB}[\texttt{MC}(C^a) \oplus \texttt{MC}(K_{10}) \oplus K_{10}] \oplus \\ & \texttt{SB}[\texttt{MC}(C^b) \oplus \texttt{MC}(K_{10}) \oplus K_{10}] \end{aligned} \tag{4.93a}$$

$$\begin{aligned} \texttt{SR}^{-1}(D^a \oplus D^c) = {} & \texttt{SB}[\texttt{MC}(C^a) \oplus \texttt{MC}(K_{10}) \oplus K_{10}] \oplus \\ & \texttt{SB}[\texttt{MC}(C^c) \oplus \texttt{MC}(K_{10}) \oplus K_{10}] \end{aligned} \tag{4.93b}$$

C^a, C^b, C^c, D^a, D^b and D^c are 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 between $(2^8)^4$ values for each column of K_{10} leads to a set of 2^8 hypotheses of column values. Repeating these operations for the three next 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 hypothesis on 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 value is revealed and the exhaustive search interrupts.

Consequently, alike this attack with memory access, a first exhaustive search of 2^{34} values continued by a second search of 2^{31} in average is required for finding the key. The main difference between two cases of having access or not to memory is on the number of required pairs of corresponding faulty and correct ciphertexts for exploiting this attack.

Scenario II - Attack 2

Attack moment: Any time between RC=0 and RC=10 FOR

Required fault model: {04}

Algorithm execution: $R_0..R_9\text{-}4\times R_{m=10..13}\text{-}R_{f=14}$

• Attack II-2 with memory access

This attack creates 4 additional rounds to normal AES execution. For more easiness, we assume `MiddleRound` and `FinalRound` as two functions that combine different operations of a middle of a final round according to equations 4.78 and 4.79. For exploiting 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:

$$C = \mathtt{FR}(M_9) \oplus K_{10} \tag{4.94}$$

$$D = \mathtt{FR}(\mathtt{MR}(\mathtt{MR}(\mathtt{MR}(\mathtt{MR}(M_9) \oplus K_{10}) \oplus 00) \oplus 00) \oplus 00) \oplus 00 \tag{4.95}$$

We remove useless operations in equation 4.95 and we rewrite it as equation 4.96:

$$\begin{aligned} D &= \mathtt{FR}(\mathtt{MR}(\mathtt{MR}(\mathtt{MR}(\mathtt{MR}(M_9) \oplus K_{10})))) \\ &= \mathtt{FR} \circ \mathtt{MR} \circ \mathtt{MR} \circ \mathtt{MR}[\mathtt{MR}(M_9) \oplus K_{10}] \end{aligned} \tag{4.96}$$

Then, by reversing `MiddleRound` and `FinalRound` operations in equation 4.96, we obtain equation 4.97:

$$\mathtt{MR}^{-1} \circ \mathtt{MR}^{-1} \circ \mathtt{MR}^{-1} \circ \mathtt{FR}^{-1}(D) = \mathtt{MR}(M_9) \oplus K_{10} \tag{4.97}$$

We rewrite the `MiddleRound` operation on right side of equation 4.97 as `MixColumns` ∘ `FinalRound` to get closer to equation 4.94 and so we obtain equation 4.98:

$$\mathtt{MR}^{-1} \circ \mathtt{MR}^{-1} \circ \mathtt{MR}^{-1} \circ \mathtt{FR}^{-1}(D) = \mathtt{MC} \circ \mathtt{FR}(M_9) \oplus K_{10} \tag{4.98}$$

We assume the new variable X as defined in equation 4.99 for more easiness:

$$X = \mathtt{FR}(M_9) \tag{4.99}$$

Then, we replace X according to its valu declared in equation 4.99 in equation 4.98 and so we get equation 4.100:

$$\mathtt{MR}^{-1} \circ \mathtt{MR}^{-1} \circ \mathtt{MR}^{-1} \circ \mathtt{FR}^{-1}(D) = \mathtt{MC}(X) \oplus K_{10} \tag{4.100}$$

Then, we perform an xor operation between equations 4.94 and 4.100. We obtain the equation 4.101:

$$C \oplus \mathtt{MR}^{-1} \circ \mathtt{MR}^{-1} \circ \mathtt{MR}^{-1} \circ \mathtt{FR}^{-1}(D) = X \oplus \mathtt{MC}(X) \tag{4.101}$$

C and D are known values.

Now, we perform an exhaustive search on the first column of X values, according to equation 4.101. The exhaustive search leads to a set of hypotheses on corresponding column of K_{10}. We repeat theses searches for 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 the hypothesis.

This two-step exhaustive search and its complexity are similar to described exploitation for 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:

$$C = \mathtt{FR}(M_9) \oplus K_{10} \tag{4.102}$$

$$D = \mathtt{FR}(\mathtt{MR}(\mathtt{MR}(\mathtt{MR}(\mathtt{MR}(M_9) \oplus K_{10}) \oplus K'_{11}) \oplus K'_{12}) \oplus K'_{13}) \oplus K'_{14} \tag{4.103}$$

$K'_{11}, K'_{12}, K'_{13}, K'_{14}$ are invalid keys with unknown values.

Exploitation of this attack needs exhaustive searches on the values of four erroneous rounds. The needed calculations cannot be done in a reasonable time. Even, by using several pairs of corresponding correct and faulty ciphertexts, we cannot reduce adequately 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:

$$C^a = \mathtt{SR} \circ \mathtt{SB}[\mathtt{MC} \circ \mathtt{SR} \circ \mathtt{SB}(M_8^a) \oplus K_9] \oplus K_{10} \tag{4.104a}$$

$$C^b = \mathtt{SR} \circ \mathtt{SB}[\mathtt{MC} \circ \mathtt{SR} \circ \mathtt{SB}(M_8^b) \oplus K_9] \oplus K_{10} \tag{4.104b}$$

$$C^c = \mathtt{SR} \circ \mathtt{SB}[\mathtt{MC} \circ \mathtt{SR} \circ \mathtt{SB}(M_8^c) \oplus K_9] \oplus K_{10} \tag{4.104c}$$

$$D^a = \mathtt{SR} \circ \mathtt{SB}(M_8^a) \oplus K_{f=9} \tag{4.105a}$$

$$D^b = \mathtt{SR} \circ \mathtt{SB}(M_8^b) \oplus K_{f=9} \tag{4.105b}$$

$$D^c = \mathtt{SR} \circ \mathtt{SB}(M_8^c) \oplus K_{f=9} \tag{4.105c}$$

If we perform an xor operation between the first equation 4.104 and each of following one, we obtain the two equations 4.106:

$$C^a \oplus C^b = \mathtt{SR} \circ \mathtt{SB}[\mathtt{MC} \circ \mathtt{SR} \circ \mathtt{SB}(M_8^a) \oplus K_9] \oplus \mathtt{SR} \circ \mathtt{SB}[\mathtt{MC} \circ \mathtt{SR} \circ \mathtt{SB}(M_8^b) \oplus K_9] \tag{4.106a}$$

$$C^a \oplus C^c = \mathtt{SR} \circ \mathtt{SB}[\mathtt{MC} \circ \mathtt{SR} \circ \mathtt{SB}(M_9^a) \oplus K_9] \oplus \mathtt{SR} \circ \mathtt{SB}[\mathtt{MC} \circ \mathtt{SR} \circ \mathtt{SB}(M_8^c) \oplus K_9] \tag{4.106b}$$

By using 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:

$$\mathtt{SR}^{-1}(C^a \oplus C^b) = \mathtt{SB}[\mathtt{MC}(D^a \oplus K_9) \oplus K_9] \oplus \mathtt{SB}[\mathtt{MC}(D^b \oplus K_9) \oplus K_9] \tag{4.107a}$$
$$\mathtt{SR}^{-1}(C^a \oplus C^c) = \mathtt{SB}[\mathtt{MC}(D^a \oplus K_9) \oplus K_9] \oplus \mathtt{SB}[\mathtt{MC}(D^c \oplus K_9) \oplus K_9] \tag{4.107b}$$

According to the distributivity property of `MixColumns`, we transform equations 4.107 to equations 4.108:

$$\mathtt{SR}^{-1}(C^a \oplus C^b) = \mathtt{SB}[\mathtt{MC}(D^a) \oplus \mathtt{MC}(K_9) \oplus K_9] \oplus \mathtt{SB}[\mathtt{MC}(D^b) \oplus \mathtt{MC}(K_9) \oplus K_9] \tag{4.108a}$$
$$\mathtt{SR}^{-1}(C^a \oplus C^c) = \mathtt{SB}[\mathtt{MC}(D^a) \oplus \mathtt{MC}(K_9) \oplus K_9] \oplus \mathtt{SB}[\mathtt{MC}(D^c) \oplus \mathtt{MC}(K_9) \oplus K_9] \tag{4.108b}$$

C^a, C^b, C^c, D^a, D^b and D^c are 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 on their first column in equations 4.108.

This exhaustive search between $(2^8)^4$ values for each column of K_9 leads to a set 2^8 hypotheses for its value. By repeating this procedure for each of three following columns, we get a new set 2^8 hypotheses for each one.

Then, a new exhaustive search is required between all the combinations of hypotheses in order to find the unique key value. In 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 for ciphering one of the plaintexts, *e.g.* M^a, in order to examine if they get 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 revealed after $(2^8)^3$ calculations in average.

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 not any difference in this attack exploitation with or without memory access.

4.2.3 Secondary Attack Scenarios

In addition to the main attack scenarios described in previous section, two other scenarios are feasible. Each of these scenarios includes two attacks targeting two consecutive encryption of the same plaintext. We describe them in this section.

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 same plaintext. On both attacks, the round counter is targeted on similar algorithm stages of two consecutive rounds.

This scenario offers more flexibility to the opponent in order to choice attack moment between almost the algorithm execution. Therefore, in this scenario, many attack moments that does not offer short and fast cryptanalytic solutions in scenario I can be exploited by using a faulty result of second similar attack on the previous or the following round.

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
$RC = 1$		$RC{\to}0$	$RC{\to}3$	$RC{\to}5$	$RC{\to}9$	$RC{\to}17$	$RC{\to}33$	$RC{\to}65$	$RC{\to}129$
	FOR					$NR{\to}1$ R_0-R_f f=17	$NR{\to}1$ R_0-R_f f=33	$NR{\to}1$ R_0-R_f f=65	$NR{\to}1$ R_0-R_f f=129
	LOOP					$NR{\to}2$ R_0-R_m-R_f m=17 f=18	$NR{\to}2$ R_0-R_m-R_f m=33 f=34	$NR{\to}2$ R_0-R_m-R_f m=65 f=66	$NR{\to}2$ R_0-R_m-R_f m=129 f=130
	ARK COMEBACK					$NR{\to}2$ R_0-R_1-R_f f=18	$NR{\to}2$ R_0-R_1-R_f f=34	$NR{\to}2$ R_0-R_1-R_f f=66	$NR{\to}2$ R_0-R_1-R_f f=130
$RC = 2$		$RC{\to}3$	$RC{\to}0$	$RC{\to}6$	$RC{\to}10$	$RC{\to}18$	$RC{\to}34$	$RC{\to}66$	$RC{\to}130$
	FOR					$NR{\to}2$ R_0-R_1-R_f f=18	$NR{\to}2$ R_0-R_1-R_f f=34	$NR{\to}2$ R_0-R_1-R_f f=66	$NR{\to}2$ R_0-R_1-R_f f=130
	LOOP					$NR{\to}3$ R_0-R_1-R_m-R_f m=18 f=19	$NR{\to}3$ R_0-R_1-R_m-R_f m=34 f=35	$NR{\to}3$ R_0-R_1-R_m-R_f m=66 f=67	$NR{\to}3$ R_0-R_1-R_m-R_f m=130 f=131
	ARK COMEBACK				$NR{\to}3$ R_0..R_2-R_f f=11	$NR{\to}3$ R_0..R_2-R_f f=19	$NR{\to}3$ R_0..R_2-R_f f=35	$NR{\to}3$ R_0..R_2-R_f f=67	$NR{\to}3$ R_0..R_2-R_f f=131

Tabe 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
$RC = 3$		RC→2	RC→1	RC→7	RC→11	RC→19	RC→35	RC→67	RC→131
	FOR				NR→3 $R_0..R_2$-R_f f=11	NR→3 $R_0..R_2$-R_f f=19	NR→3 $R_0..R_2$-R_f f=35	NR→3 $R_0..R_2$-R_f f=67	NR→3 $R_0..R_2$-R_f f=131
	LOOP				NR→4 $R_0..R_2$-R_m-R_f m=11 m=12	NR→4 $R_0..R_2$-R_m-R_f m=19 f=20	NR→4 $R_0..R_2$-R_m-R_f m=35 f=36	NR→4 $R_0..R_2$-R_m-R_f m=67 f=68	NR→4 $R_0..R_2$-R_m-R_f m=131 f=132
	ARK COMEBACK				NR→4 $R_0..R_3$-R_f f=12	NR→4 $R_0..R_3$-R_f f=20	NR→4 $R_0..R_3$-R_f f=36	NR→4 $R_0..R_3$-R_f f=68	NR→4 $R_0..R_3$-R_f f=132
$RC = 4$		RC→5	RC→6	RC→0	RC→12	RC→20	RC→36	RC→68	RC→132
	FOR				NR→4 $R_0..R_3$-R_f f=12	NR→4 $R_0..R_3$-R_f f=20	NR→4 $R_0..R_3$-R_f f=36	NR→4 $R_0..R_3$-R_f f=68	NR→4 $R_0..R_3$-R_f f=132
	LOOP				NR→5 $R_0..R_3$-R_m-R_f m=12 f=13	NR→5 $R_0..R_3$-R_m-R_f m=20 f=21	NR→5 $R_0..R_3$-R_m-R_f m=36 f=37	NR→5 $R_0..R_3$-R_m-R_f m=68 f=69	NR→5 $R_0..R_3$-R_m-R_f m=132 f=133
	ARK COMEBACK				NR→5 $R_0..R_4$-R_f f=13	NR→5 $R_0..R_4$-R_f f=21	NR→5 $R_0..R_4$-R_f f=37	NR→5 $R_0..R_4$-R_f f=69	NR→5 $R_0..R_4$-R_f f=133

Tabe continued on next page

Table 4.6 – Table continued from previous page

Round	Attack	Fault model							
counter	moment	0x01	0x02	0x04	0x08	0x10	0x20	0x40	0x80
$RC = 5$		$RC{\to}4$	$RC{\to}7$	$RC{\to}1$	$RC{\to}13$	$RC{\to}21$	$RC{\to}37$	$RC{\to}69$	$RC{\to}133$
	FOR				$NR{\to}5$ $R_0..R_4$-R_f f=13	$NR{\to}5$ $R_0..R_4$-R_f f=21	$NR{\to}5$ $R_0..R_4$-R_f f=37	$NR{\to}5$ $R_0..R_4$-R_f f=69	$NR{\to}5$ $R_0..R_4$-R_f f=133
	LOOP				$NR{\to}6$ $R_0..R_4$-R_m-R_f m=13 f=14	$NR{\to}6$ $R_0..R_4$-R_m-R_f m=21 f=22	$NR{\to}6$ $R_0..R_4$-R_m-R_f m=37 f=38	$NR{\to}6$ $R_0..R_4$-R_m-R_f m=69 f=70	$NR{\to}6$ $R_0..R_4$-R_m-R_f m=133 f=134
	ARK COMEBACK				$NR{\to}6$ $R_0..R_5$-R_f f=14	$NR{\to}6$ $R_0..R_5$-R_f f=22	$NR{\to}6$ $R_0..R_5$-R_f f=38	$NR{\to}6$ $R_0..R_5$-R_f f=70	$NR{\to}6$ $R_0..R_5$-R_f f=134
$RC = 6$		$RC{\to}7$	$RC{\to}4$	$RC{\to}2$	$RC{\to}14$	$RC{\to}22$	$RC{\to}38$	$RC{\to}70$	$RC{\to}134$
	FOR				$NR{\to}6$ $R_0..R_5$-R_f f=14	$NR{\to}6$ $R_0..R_5$-R_f f=22	$NR{\to}6$ $R_0..R_5$-R_f f=38	$NR{\to}6$ $R_0..R_5$-R_f f=70	$NR{\to}6$ $R_0..R_5$-R_f f=134
	LOOP				$NR{\to}7$ $R_0..R_5$-R_m-R_f m=14 f=15	$NR{\to}7$ $R_0..R_5$-R_m-R_f m=22 f=23	$NR{\to}7$ $R_0..R_5$-R_m-R_f m=38 f=39	$NR{\to}7$ $R_0..R_5$-R_m-R_f m=70 f=71	$NR{\to}7$ $R_0..R_5$-R_m-R_f m=134 f=135
	ARK COMEBACK				$NR{\to}7$ $R_0..R_6$-R_f f=15	$NR{\to}7$ $R_0..R_6$-R_f f=23	$NR{\to}7$ $R_0..R_6$-R_f f=39	$NR{\to}7$ $R_0..R_6$-R_f f=71	$NR{\to}7$ $R_0..R_6$-R_f f=135

Tabe 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
$RC = 7$		$RC{\to}6$	$RC{\to}5$	$RC{\to}3$	$RC{\to}15$	$RC{\to}23$	$RC{\to}39$	$RC{\to}71$	$RC{\to}135$
	FOR				$NR{\to}7$ $R_0..R_6$-R_f f=15	$NR{\to}7$ $R_0..R_6$-R_f f=23	$NR{\to}7$ $R_0..R_6$-R_f f=39	$NR{\to}7$ $R_0..R_6$-R_f f=71	$NR{\to}7$ $R_0..R_6$-R_f f=135
	LOOP				$NR{\to}8$ $R_0..R_6$-R_m-R_f m=15 f=16	$NR{\to}8$ $R_0..R_6$-R_m-R_f m=23 f=24	$NR{\to}8$ $R_0..R_6$-R_m-R_f m=39 f=40	$NR{\to}8$ $R_0..R_6$-R_m-R_f m=71 f=72	$NR{\to}8$ $R_0..R_6$-R_m-R_f m=135 f=136
	ARK COMEBACK				$NR{\to}8$ $R_0..R_7$-R_f f=16	$NR{\to}8$ $R_0..R_7$-R_f f=24	$NR{\to}8$ $R_0..R_7$-R_f f=40	$NR{\to}8$ $R_0..R_7$-R_f f=72	$NR{\to}8$ $R_0..R_7$-R_f f=136
$RC = 8$		$RC{\to}9$	$RC{\to}10$	$RC{\to}12$	$RC{\to}0$	$RC{\to}24$	$RC{\to}40$	$RC{\to}72$	$RC{\to}136$
	FOR			$NR{\to}8$ $R_0..R_7$-R_f f=12		$NR{\to}8$ $R_0..R_7$-R_f f=24	$NR{\to}8$ $R_0..R_7$-R_f f=40	$NR{\to}8$ $R_0..R_7$-R_f f=72	$NR{\to}8$ $R_0..R_7$-R_f f=136
	LOOP			$NR{\to}9$ $R_0..R_7$-R_m-R_f m=12 f=13		$NR{\to}9$ $R_0..R_7$-R_m-R_f m=24 f=25	$NR{\to}9$ $R_0..R_7$-R_m-R_f m=40 f=41	$NR{\to}9$ $R_0..R_7$-R_m-R_f m=72 f=73	$NR{\to}9$ $R_0..R_7$-R_m-R_f m=136 f=137
	ARK COMEBACK		$NR{\to}9$ $R_0..R_8$-R_f f=11	$NR{\to}9$ $R_0..R_8$-R_f f=13		$NR{\to}9$ $R_0..R_8$-R_f f=25	$NR{\to}9$ $R_0..R_8$-R_f f=41	$NR{\to}9$ $R_0..R_8$-R_f f=73	$NR{\to}9$ $R_0..R_8$-R_f f=137

Tabe 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
$RC = 9$		$RC \to 8$	$RC \to 11$	$RC \to 13$	$RC \to 1$	$RC \to 25$	$RC \to 41$	$RC \to 73$	$RC \to 137$
	FOR		$NR \to 9$ $R_0..R_8$-R_f f=11	$NR \to 9$ $R_0..R_8$-R_f f=13		$NR \to 9$ $R_0..R_8$-R_f f=25	$NR \to 9$ $R_0..R_8$-R_f f=41	$NR \to 9$ $R_0..R_8$-R_f f=73	$NR \to 9$ $R_0..R_8$-R_f f=137
	LOOP		$NR \to 10$ $R_0..R_8$-R_m-R_f m=11 f=12	$NR \to 10$ $R_0..R_8$-R_m-R_f m=13 f=14		$NR \to 10$ $R_0..R_8$-R_m-R_f m=25 f=26	$NR \to 10$ $R_0..R_8$-R_m-R_f m=41 f=42	$NR \to 10$ $R_0..R_8$-R_m-R_f m=73 f=74	$NR \to 10$ $R_0..R_8$-R_m-R_f m=137 f=138
	ARK COMEBACK		$NR \to 10$ $R_0..R_9$-R_f f=12	$NR \to 10$ $R_0..R_9$-R_f f=14		$NR \to 10$ $R_0..R_9$-R_f f=26	$NR \to 10$ $R_0..R_9$-R_f f=42	$NR \to 10$ $R_0..R_9$-R_f f=74	$NR \to 10$ $R_0..R_9$-R_f f=138
$RC = 10$		$RC \to 11$	$RC \to 8$	$RC \to 14$	$RC \to 2$	$RC \to 26$	$RC \to 42$	$RC \to 74$	$RC \to 138$
	FOR LOOP	$NR \to 10$ $R_0..R_9$-R_f f=11		$NR \to 10$ $R_0..R_9$-R_f f=14		$NR \to 10$ $R_0..R_9$-R_f f=26	$NR \to 10$ $R_0..R_9$-R_f f=42	$NR \to 10$ $R_0..R_9$-R_f f=74	$NR \to 10$ $R_0..R_9$-R_f f=138

4.2.3.2 Double Attack on Round Counter Value and 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 two different moments:

1st moment option: $RC{=}n$ ARK, COMEBACK or $RC{=}n+1$ FOR
$2 \leq$ n ≤ 8 ; also feasible for n=1 and n=9, but without any advantage.
Required fault model: {10, 20, 40, 80} ∪ {other corresponding fault as mentioned on appendix A}
Algorithm execution: $R_0..R_n$-R_{f1}

or

2nd moment option: $RC{=}n$ LOOP
$2 \leq$ n ≤ 8 ; also feasible for n=1 and n=9, but without any advantage.
Required fault model: {10, 20, 40, 80} ∪ {other corresponding fault as mentioned on appendix A}
Algorithm execution: $R_0..R_{n-1}$-R_{m1}-R_{f1}

For the second attack also, two different moment can be considered:

1st moment option: $RC{=}n+1$ LOOP
Required fault model: {10, 20, 40, 80} ∪ {other corresponding fault as mentioned on 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} ∪ {other corresponding fault as mentioned on appendix A}
Algorithm execution: $R_0..R_{n+1}$-R_{f2}

Therefore, each of four possible combination of a 1st and a 2nd attack leads to a special cryptanalysis solution exploiting the differential round values.

4.3 Conclusions

In this chapter, we extended our research to explore new models of fault attacks. Fault attacks on the round counter or on the reference number of rounds have been much less studied than 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. A dozen of these attacks is based on simple scenarios, including an attack at a specific time of algorithm execution, on one of the two aforementioned targets.

Therefore, the implementation of appropriate countermeasures is necessary for this kind of attacks targeting the execution of a repetitive cryptographic algorithm.

Chapter 5

Countermeasures

Contents

5.1 Introduction

In chapters 3 and 4, we have reported two attack schemes on cryptosystems by laser fault injection. These attacks may menace security of circuit contents and provide access to secrets for opponents. 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 them. Personnel security and also physical structures, areas, and devices may be intended as 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 survey a solution against 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 be also very appreciable if this countermeasure may safeguard the circuit against other means of fault attacks. However, this desired countermeasure depends to layout design issues and researches. So, it is out of current thesis' scope. We survey in this chapter only on embedded techniques and protections against these attacks. Besides, this presumptive hardware solution may be vulnerable by upcoming etching methods or surgical laser fault injection techniques. To this end, even by using hardware countermeasure, the embedded protections are strongly needed.

A well designed embedded countermeasure against fault attacks on the round keys must detect immediately any change in their values and prevents the release of faulty ciphertext. Therefore, faulty results containing secret information will be hidden from the opponent and any possibility no longer remains for DFA.

5.2.1 A Countermeasure against Fault Attacks on the Round Keys

We described in section 2.2 that in our AES implementation all the round keys are derived for 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 had added two security transformations:

- `Kxoring` operation to `KeyExpansion`
- `KeyTest` operation to `DataEncryption`

`Kxoring` is based on information redundancy: bitwise parity regarding the columns. It computes $KxorRef$ for each derived round key during `KeyExpansion` processes. Then, in `DataEncryption` processes, `KeyTest` calculates $Kxor$ value for last used round key after each `AddRoundKey`. `KeyTest` checks the integrity of the used round key by comparing current $Kxor$ to corresponding $KxorRef$.

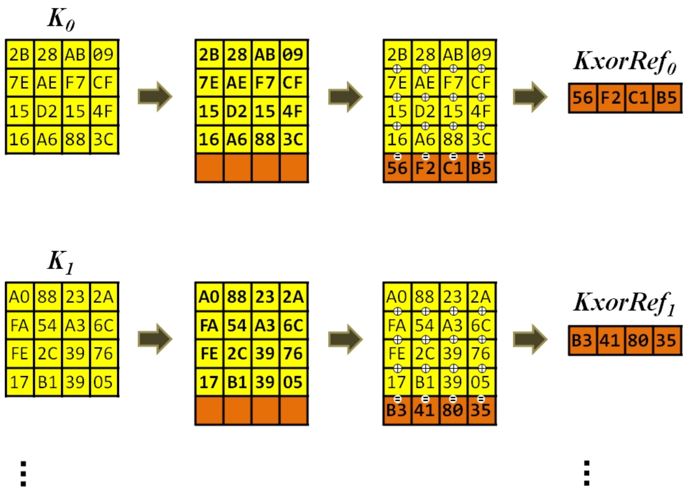


Figure 5.1: Calculation of `KxorRef` bytes.

$KxorRef$ is an array of 11 blocs, similarly to the number of round keys. Each bloc is composed of 4 bytes. Each byte stores the xor product of corresponding subkey 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 following operations:

```
KxorRef[i][0] ← K[i][0] ⊕ K[i][1] ⊕ K[i][2] ⊕ K[i][3]
KxorRef[i][1] ← K[i][4] ⊕ K[i][5] ⊕ K[i][6] ⊕ K[i][7]
KxorRef[i][2] ← K[i][8] ⊕ K[i][9] ⊕ K[i][10] ⊕ K[i][11]
KxorRef[i][3] ← K[i][12] ⊕ K[i][13] ⊕ K[i][14] ⊕ K[i][15]
```

$KxorRef$ 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 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 below instructions:

```
C ← C ⊕ K[RC]
```

```
    IF (Kxor(K[RC]) != KxorRef[RC]) THEN
        C ← 0
        BREAK
    ENDIF
```

Any difference indicates existence of a fault on the current round key. Therefore, the countermeasure takes appropriate action and returns a fully zero ciphertext 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 not any 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. But, 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. Therefore, 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 to 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. We examine by the `KeyCompare` operation, if reversed `AddRoundKey` operations lead to the secret key. Once this test is passed, the integrity of stored round keys

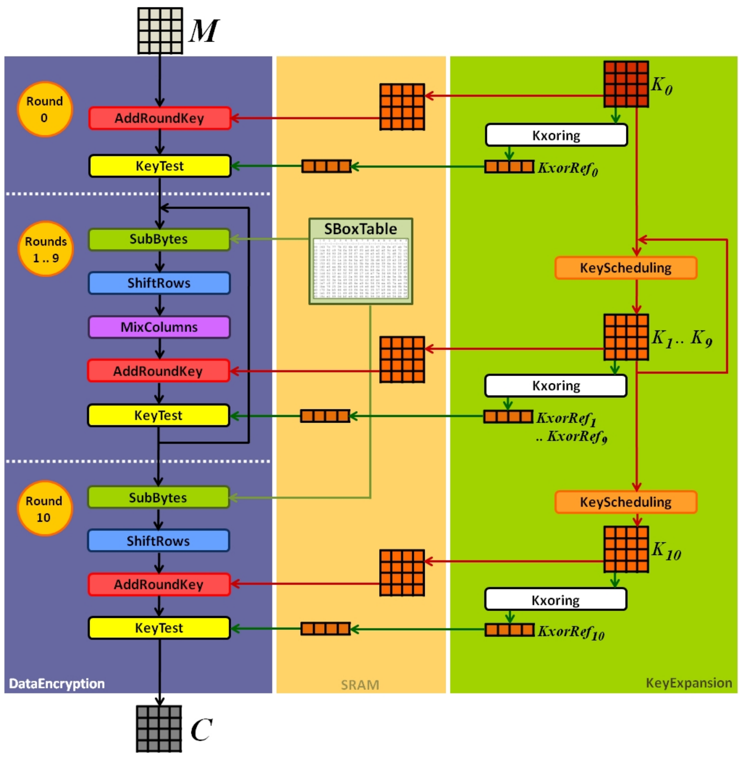


Figure 5.2: Key-protection countermeasure by `Kxoring` calculation and `KeyTest` operation in our AES implementation.

is controlled by `KeyTest` countermeasure during all the 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 is 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 protects 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 on the chip's ALU or 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 final round. The secondary encryption is done after end of the first one. Then, both of final ciphertexts are compared by `CipherTest` operation at the end

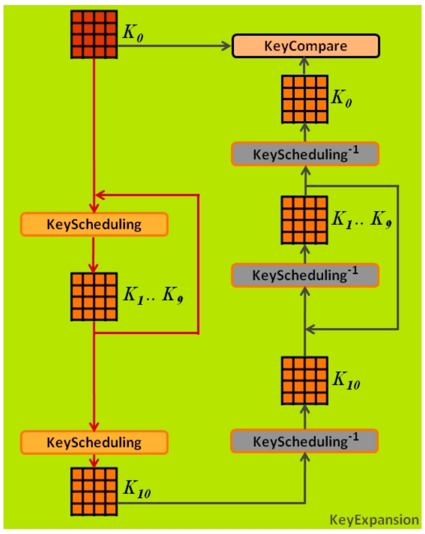


Figure 5.3: `KeyCompare` countermeasure.

of second encryption. If any difference between two final ciphertexts is detected, `CipherTest` changes the ciphertext value to 0 and neutralize 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. So, we duplicated `SubBytes` look-up table for the refers from secondary encryption path.

We implemented these multiple countermeasures on 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 any more retrieve the faulty ciphertexts. These solutions guarantee the required security against DFA in context of our attacks and protect 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 cannot unfortunately increase the hardware resources or create parallel hardware dataflows.

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.

At the first step, we added `Kxoring` operations and reverse of `KeyScheduling` to the standard `KeyExpansion`. These operations are done just once and have not

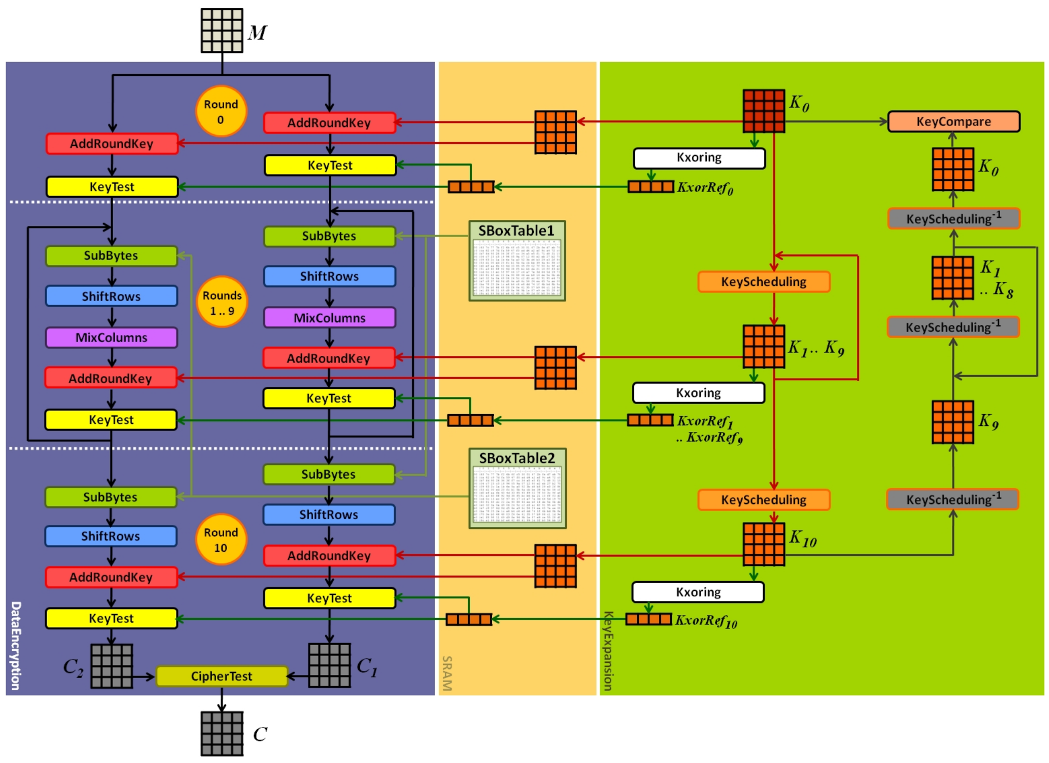


Figure 5.4: Implementation of parallel encryption processes and `KeyTest` and `KeyCompare` operations in our AES implementation.

any effect on the `DataEncyption` processing time.

Our countermeasures increased the `DataEncryption` operations by adding `KeyTest` and duplication of all the operations. Consequently, the encryption time of each text is increased for more than 110%. Therefore, we surveyed for lighter solutions without waiving the security requirements.

According to the picture 3.6 and 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 and upcoming DFA attack, we expand the required protecting area to an additional round. So, we focus on protecting the `DataEncryption` only from `MixColumns` exit of R_6 until 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 end of R_{10}. Then, if the two ciphertexts are equal, they will be released; otherwise, they will be set to `0`.

Figure 5.5 shows implementation of our improved countermeasures. In our new

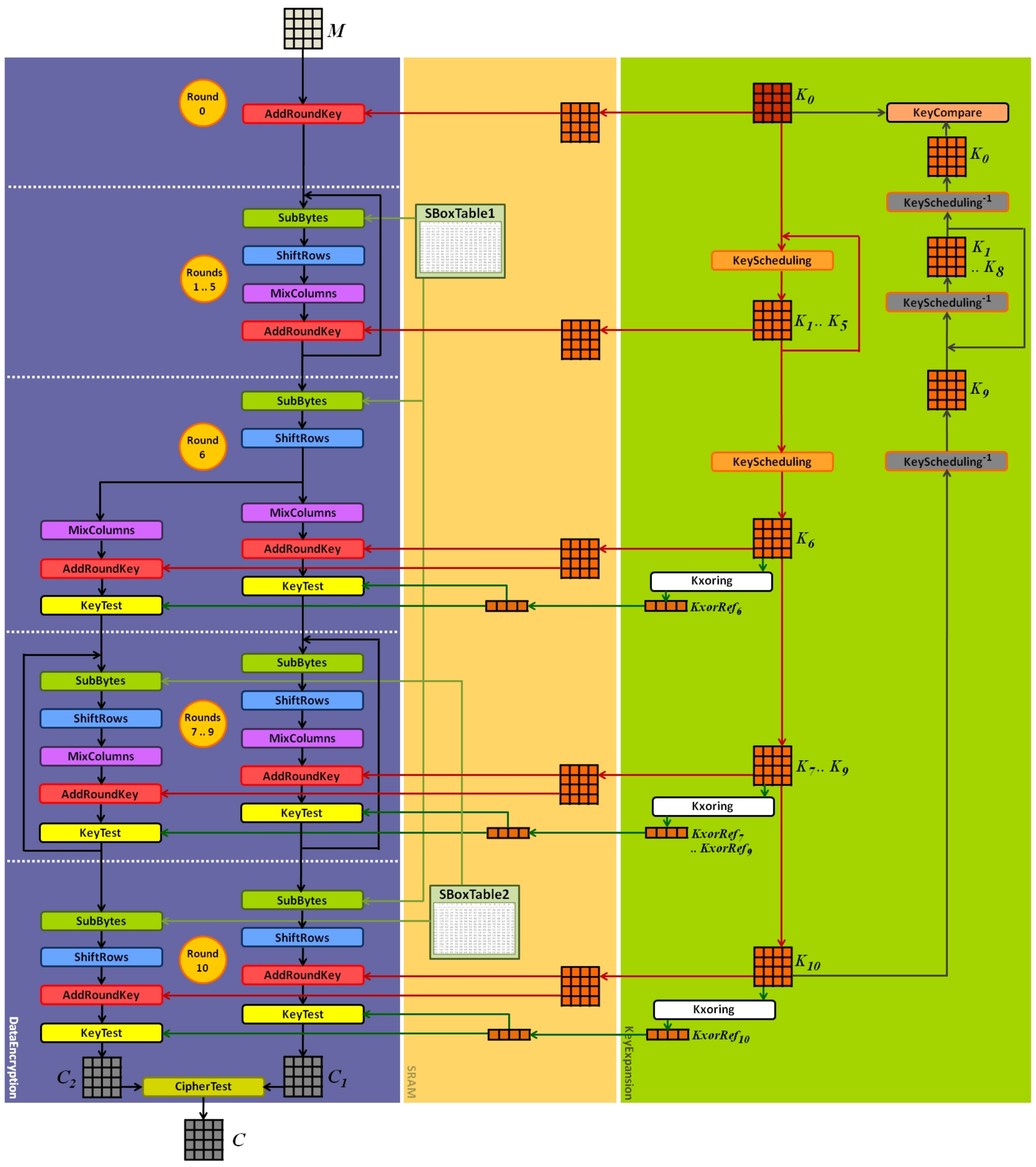


Figure 5.5: Implementation of the improved countermeasures in our AES implementation.

experiment they provide the required security against DFA in shorter execution time in comparison 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 round reduction or addition using faults. Now, in the current section, we present appropriate countermeasures against this kind 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 these two values can be targeted by a laser fault attack. Then, the opponent may discover the key value through the faulty ciphertexts using appropriate cryptanalysis.

We need a solution that protects our circuit against any round modification by laser fault injection on RC and R_{max}. We surveyed on possibility of removing these two potential targets. Therefore, we unrolled AES middle rounds implementation in order to each round is 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 not any more change neither the round in execution, nor the number of total executed rounds. Nevertheless, it seems that an attack on the *program counter* may cause a jump in the algorithm and realize again a round modification attack.

In our experiments, we did not have any success to change the program counter by laser attack. However, we did not ignore an eventual fault attack on the program counter. So, we improved our algorithm in order to resist against any eventual attack that may change the program counter.

5.3.2 A More Secure Unrolled AES Countermeasure

An AES algorithm that resists against any round modification attack, must perform only 10 rounds after the initial round and check 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 left at each executing round until end of 6-th round. The second counter is initialized at 128 or `0x80`, then it is shifted one bit to right from 6-th round. Therefore in this algorithm, we have two separated counters that just monitored and not conducted the correct execution before returning the ciphertext.

For having 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 and so 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 of round counters, it will be almost impossible to change all the three counters, *i.e.* RC, $Counter_1$ and $Counter_2$ 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
```

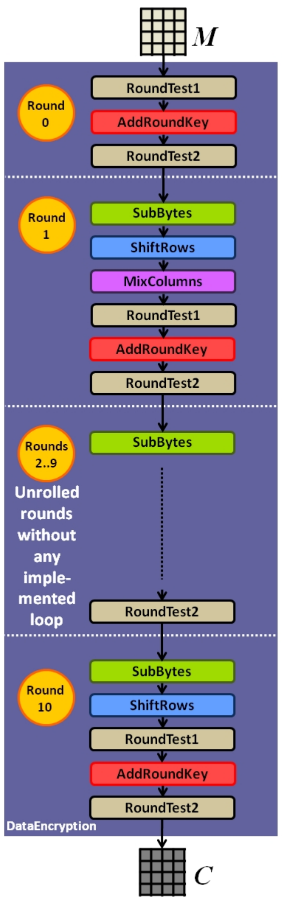


Figure 5.6: Implementation of the unrolled AES with `RoundTest1` and `RoundTest2` operations.

```
            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 $\leftarrow$ SBOX(C)
    C $\leftarrow$ SR(C)
    IF ((Counter1=128) & (Counter2=16)) THEN
        C $\leftarrow$ C $\oplus$ K[10]
        Counter2 $\gg$ 1
    ENDIF
//END OF ENCRYPTION
    IF ((Counter1=128) & (Counter2=8)) THEN
        RETURN C
    ELSE
        C $\leftarrow$ 0
        RETURN C
    ENDIF
END
```

This countermeasure detects any round modification via its internal monitoring counters. If any round modification or jump happens, it returns a fully zero ciphertext.

Therefore, by using theses 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 attacks and RMA by faults. We can combine these two set of countermeasures in order to obtain a full protection against DFA attacks and RMA by faults.

Figure 5.7 shows an overview of the combined countermeasure against DFA attacks and RMA in our AES implementation. We consider `KeyTest` as 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 attacks and RMA 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 that are usually considered, use and implementation of such countermeasures are mandatory in order to protect cryptographic circuits.

We expect that the presented embedded countermeasures protect strongly our AES and can be adapted easily to other implementations in order to protect them.

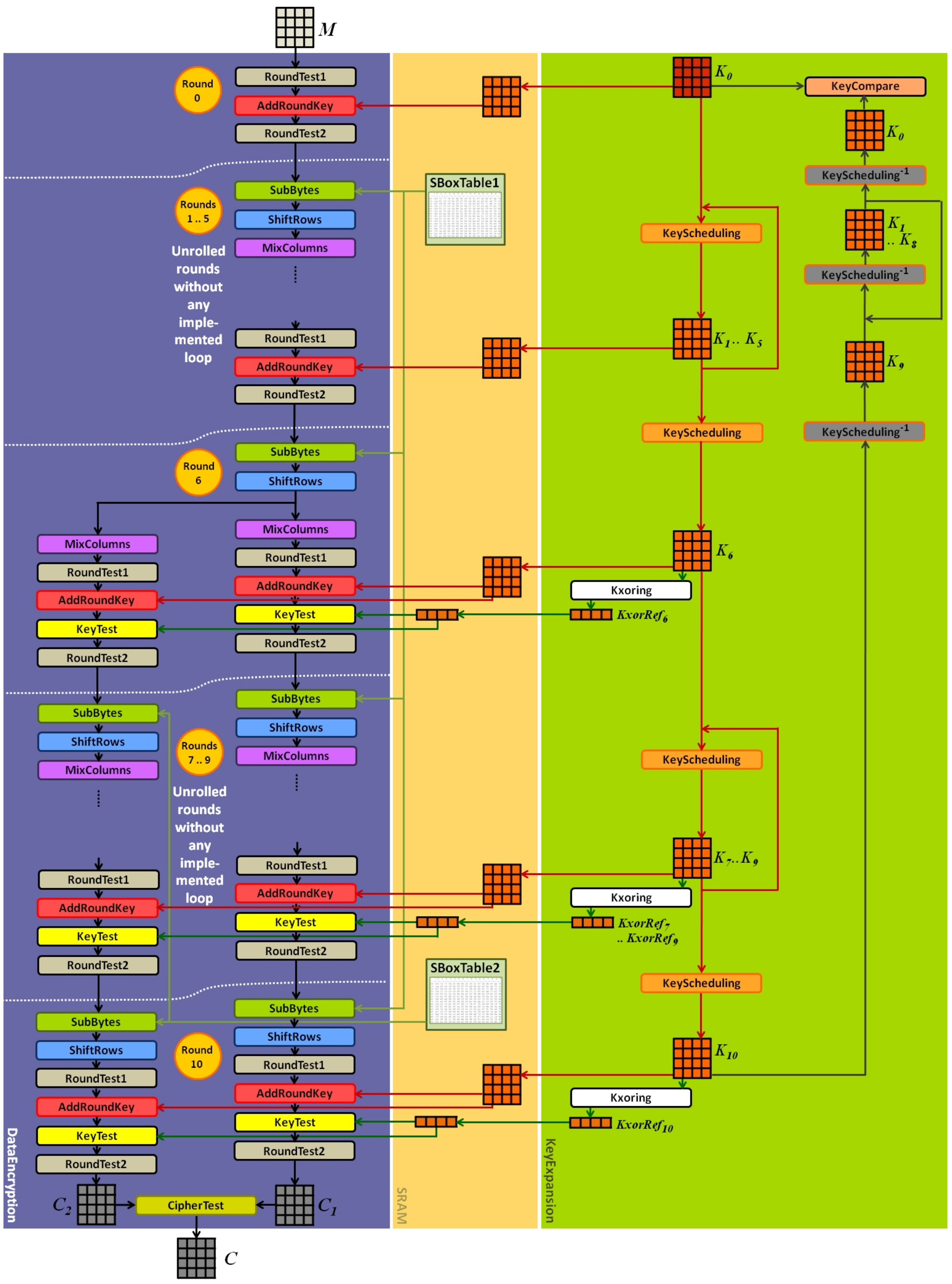


Figure 5.7: Combined countermeasures against DFA attacks and RMA in our AES implementation.

Nevertheless, presented countermeasures take more circuit resources and increase the encryption time. However, using these countermeasures is a proper trade-off in order to protect sensitive information in an AES cryptosystem by losing some execution time and memory space efficiencies.

Chapter 6

Other Security Perspectives

Contents

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 is to make the 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 on using an embedded set 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 in the plastic shape of upcoming smart card generations.

6.2.1 Introduction

In section 1.1, we described *Side Channel Analysis* (SCA) as a principle category in hardware attacks. They consists in using any hardware information leakage during the encryption operations to discover secrets from it [Kocher 1998] [Kocher 1999]. Within these attacks, Single and Differential Power Analysis (SPA and DPA) are particular analysis techniques exploiting any power consumption variation to discover the operations executed by the hardware. Many countermeasures have been developed against these attacks, however their efficiency is not still optimal.

6.2.2 Criteria for a Perfect Solution

While the power supply line of a smart card chip is connected to its external contact, any countermeasure cannot hide entirely the power consumption variations from the attacker. The only solution is to separate the power supply line from its contact. A means to meet this requirement is to provide a separated power supply in the card structure. The embedded power supply will need an external power source that provides its energy by the card contact, but it should mask entirely the chip's consumption. It is difficult to find such an electrical source that could fit in the smart card structure.

A set of a light source and a photovoltaic cell could be a choice, if they can satisfy the size, the power and the fast switching requirements for a smart card. Between the light sources, a *light-emitting diode* (LED) or an *organic* LED (OLED) could be good choices to provide these criteria.

6.2.2.1 LED

A light-emitting diode (LED) is a semiconductor diode that emits light when 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 visible domain or invisible (*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 and subject of many researches for this aim.

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 are some researches to use OLEDs as light sources for general lighting and the first products are being commercialized.

6.2.2.3 Photovoltaic Cell

A photovoltaic cell is a device composed of semiconductor layers that 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 and they flow through the semiconducting material. Due to the special composition of photovoltaic cells, the electrons are only allowed to move in a single direction. So, they create an electrical potential difference between two edges of the photovoltaic cell that corresponds to the current direction. The produced electricity can be obtained as a direct current from the two edges.

Current technologies of commercialized photovoltaic cell have an efficiency of about 20%. Besides, different researches achieved to discover new materials with an efficiency that sometimes reaches above 40%. The most efficient photovoltaic cells are often used for satellite or military purposes and are not available in the 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 power supply for the smart card's microcontroller. So, without any change on the card reader, it still will provide the electrical power to the card's contact. But, in the new card structure, the V_{cc} line must be connected only to the OLED panel. Then, the electrical energy for the smart card's microcontroller will be supplied by the photovoltaic cell .

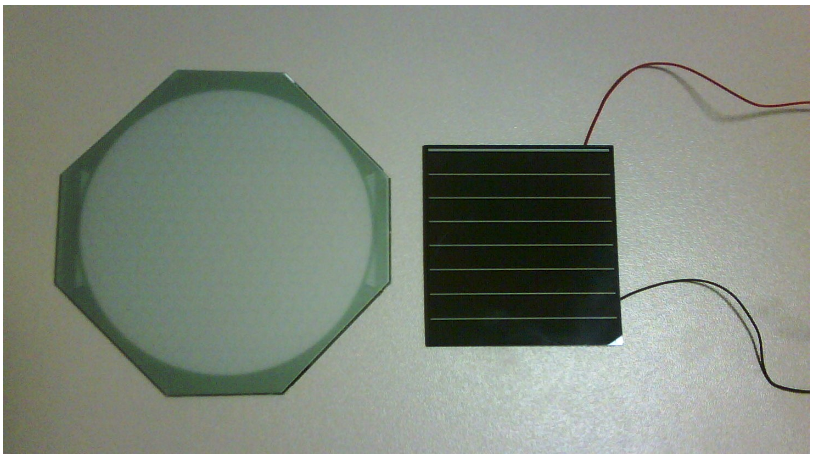


Figure 6.1: Osram Orbeos OLED and Sanyo Amorton AM-8801.

For implementing our solution, we searched between available technologies on the market. For the light emitter, we were looking for a solution in smart card size with a good efficiency. We found many single and array LEDs, some OLED displays for mobile and PDAs and only another one available as a commercialized OLED panel for illumination: The Osram Orbeos CDW-031 OLED [Semiconductors 2009]. It is a thin panel as shown on left side of figure 6.1. The OLED layer without its protecting glass has a thickness of only several hundred nanometers. Some of its technical informations are reported on table 6.1 [Semiconductors 2009]. We also chose some LED arrays for our experiments.

Table 6.1: Some technical informations about Osram Orbeos CDW-031 OLED panel.

Parameter	Value	Unit
Diameter of light output area	79	mm
Forward voltage (Max.)	4.5	V
Power consumption (Max.)	0.71	W
Luminance (for a forward current of 186 mA)	1000	Cd/m^2
Luminous efficacy (typical value)	23	lm/W

For the photovoltaic cell, we found many models. But, by applying our criteria for a size proper to LED arrays and OLED panel surfaces and for a good efficiency, our choices became very limited. We selected some models as described on table 6.2 for their higher power values. For two models that have small dimensions, we obtained a quantity of cells that cover the LED arrays or the OLED's light output surface.

For finding best photovoltaic cell, at the first, we tested output of different cells by LED arrays and OLED panel. As the results showed a better efficiency with the OLED panel, we focused on it for the next 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 at its maximum authorized power as the first level measurement and then boosted it between 2 and 8.5 times higher for the following levels. To obtain more precise results, we applied 9 different resistance levels from 470 Ω 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 (shown on the right side of figure 6.1) for its better efficiency [Sanyo 2008].

Boosting the OLED panel during the test has always increased the output, but the efficiency did not grow after the third power level and even it decreases quickly. Figure 6.2 shows the output and the efficiency curves by applying a 1000Ω resistance at the output of AM-8801 photovoltaic cell.

Table 6.2: Maximum output power of each photovoltaic cell model when exposed to Osram Orbeos OLED panel light, boosted at $8.25W$.

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	LS60×60-4M150	50.0 × 50.0	1	8.17	470

Therefore, we implemented our countermeasure by supplying V_{cc} of the card's microcontroller directly form the photovoltaic cell. The current supplied by the photovoltaic cell could successfully turn on generic smart cards that their supply voltage can vary from 3 to $5V$. But, it did not have sufficient stability to turn on an EMV (Eurocard, Mastercard, Visa) chip card.

According to [ISO/IEC 7816-3 2003], the EMV cards have a protection against voltage variations that disables them when the V_{cc} is more than 10% below or above than its typical value . So, we used an Infineon TLE 4264 as a $5V$ voltage regulator to maintain the voltage stability. It needs to two capacitors at entry and exit to establish a stable voltage of $5V$. We used a 100μF electrolytic capacitor at input and a 10μF electrolytic capacitor at output for testing with different cards. Figure 6.3 shows a scheme of our countermeasure.

We can consider the set of OLED panel, photovoltaic cell, voltage regulator, two capacitors and the card's chip as a *protected system* that must hide power consumption variations from its external world. We built a card adapter by soldering the protected system elements and a smart card connector on a Krystal Universal Card, as shown in figure 6.6. Now, when we insert a smart card on it, the set demonstrates the prototype of a future generation secure card against power analysis attacks.

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 are taken by two recent EMV chip cards supplied by GyD Iberica under the reference 05/09 16953410 and by Sagem Orga under the reference 04/10 103043-1.

We boosted the OLED panel input power to about 3.5 times higher than its maximum authorized power 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 7816-3 permits a current until $60mA$ as maximum and a voltage that could be until 10% higher than the typical $5V$. However, as the older cards have higher power consumption, they could not turned on by the limited energy of our photovoltaic cell. For these cards, we boosted the OLED panel to at about 16 times higher than its maximum authorized power. Table 6.3 shows these results.

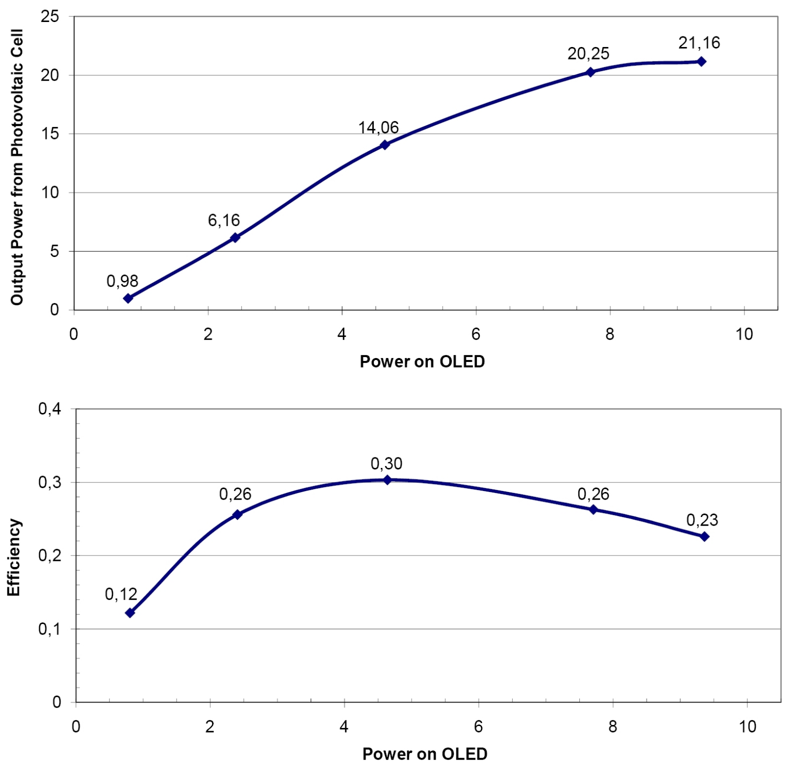


Figure 6.2: Output and efficiency of the system by boosting the OLED panel over its authorized power limits.

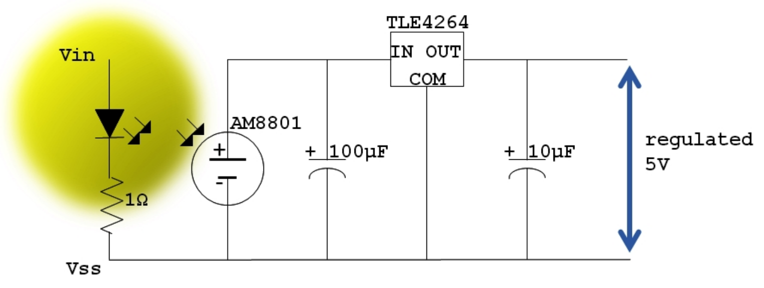


Figure 6.3: Electronic scheme of the countermeasure.

Table 6.3: Results for different smart cards using 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	✓

For performing our tests, we used a Smartware X-CORE T series card reader. The reset commands are sent by SMarT-I (described in subsection 2.1.1) to the card reader.

Figure 6.8 shows that after applying our countermeasure, power consumption curve on V_{cc} contact of the reader is entirely flat and does not show any variation. So, any power analysis attack on the reader become impossible.

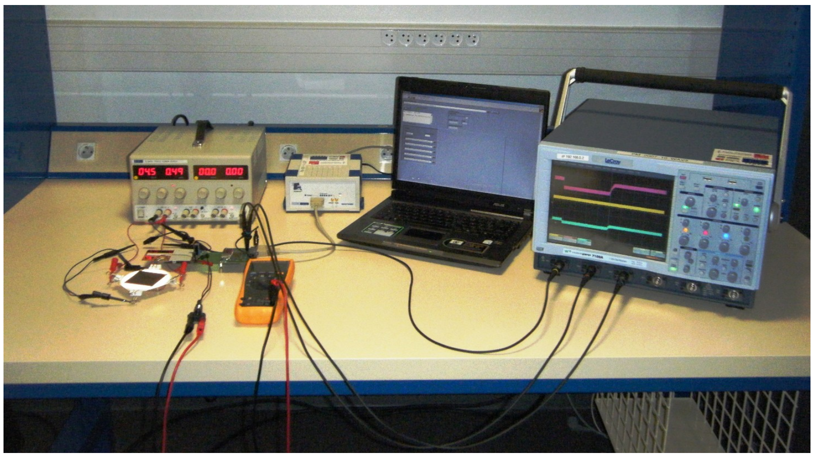


Figure 6.4: The test bench.

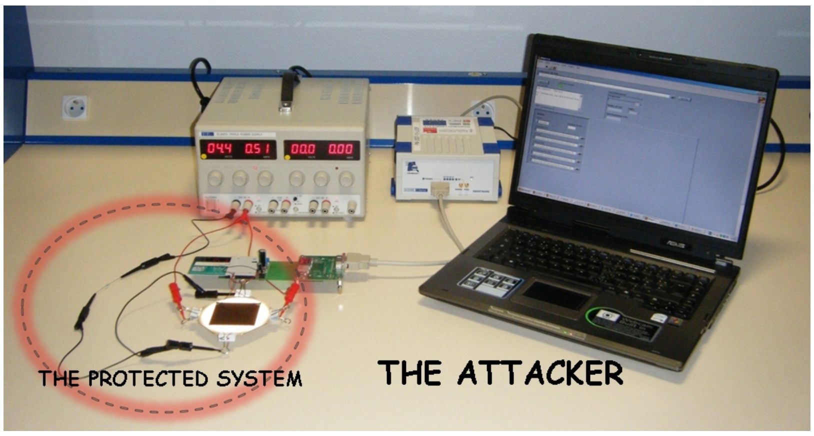


Figure 6.5: The protected system and the attacker.

The minimum value of the input capacitor is a function of photovoltaic output power and card consumption. For GyD Iberica card, it can reduce to 50μF to answer successfully to a single reset command. But, in the case of sending immediately a second reset command, the voltage regulator could not maintain the regulated voltage at 5V and card did not respond. To send continuously commands, the minimum value for this capacitor in our tests is measured 70μF.

More experiences showed that the necessary input power on OLED can be reduced by using higher value for the input capacitor; *e.g.* by using two capacitors of 100μF instead of only one for Gemplus 04/03 46156, the card needs an input power of 3.29mW on the OLED instead of initial value of 4.13mW.

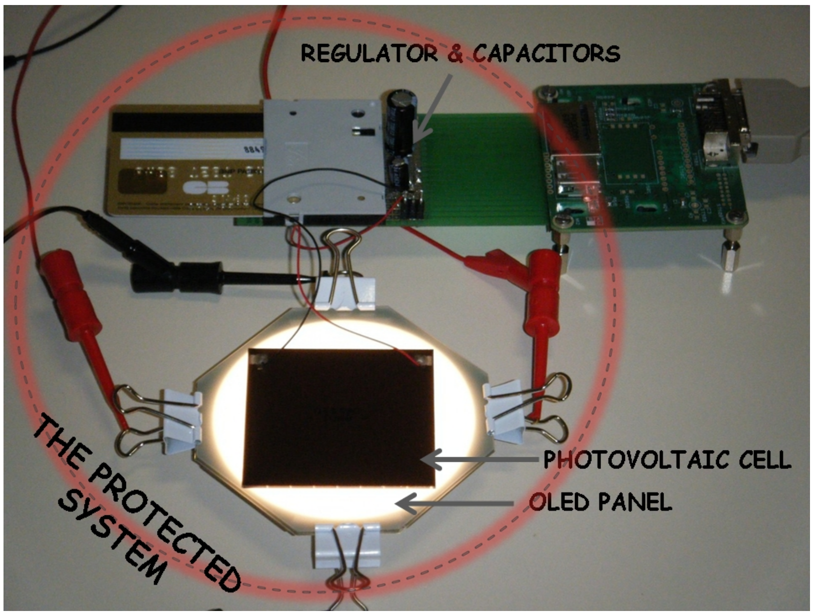


Figure 6.6: Closeup on the protected system.

So, we could build our countermeasure against power analysis attacks. As the set of OLED panel and the photovoltaic cell will have a thickness of few hundred microns without their protecting glasses, they can be integrated on the plastic shape of oncoming generations of smart cards. In this case, the energy wasting in the glasses will be also discarded. Figure 6.7 shows a close-up on the thickness of OLED panel and photovoltaic cell with their protecting glasses and a smart card.

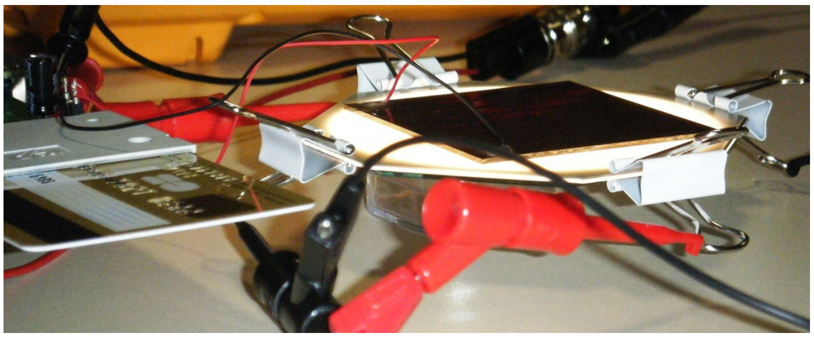
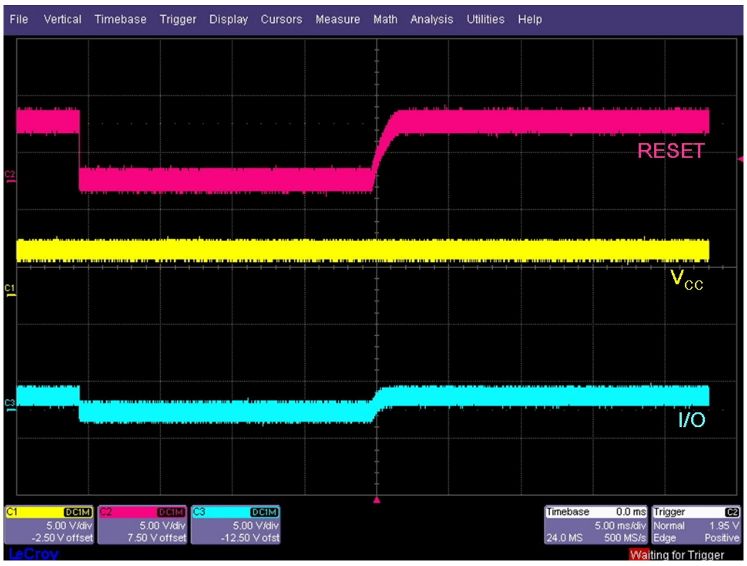


Figure 6.7: Close-up on the thicknesses of OLED and photovoltaic cell (both with their protecting glasses).



```
C1 DC1M   5.00 V/div  -2.50 V offset
C2 DC1M   5.00 V/div   7.50 V offset
C3 DC1M   5.00 V/div  -12.50 V ofst
Timebase  0.0 ms   5.00 ms/div   24.0 MS   500 MS/s
Trigger C2   Normal  1.95 V   Edge  Positive
Waiting for Trigger
```

Figure 6.8: Reset, power consumption and I/O curves 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 see the chip's consumption curves using an embedded resistance in the card reader. 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 to avoid connecting two ground lines. As shown in figure 6.9, we considered a twisted pair connection between I/O and clock contacts of the card and the reader. The principle is like the implementation of two data lines namely `D+` and `D-` in an Universal Serial Bus (USB) to establish a direct data transfer connection between two components without using a common ground line.

6.2.5 Conclusion

We demonstrated our solution against power analysis attacks on smart cards. They can powered on indirectly using an OLED panel and a photovoltaic cell. So, 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 plastic shape for oncoming generations of smart cards.

Our countermeasure has a power consumption about 7.6 times greater than ISO

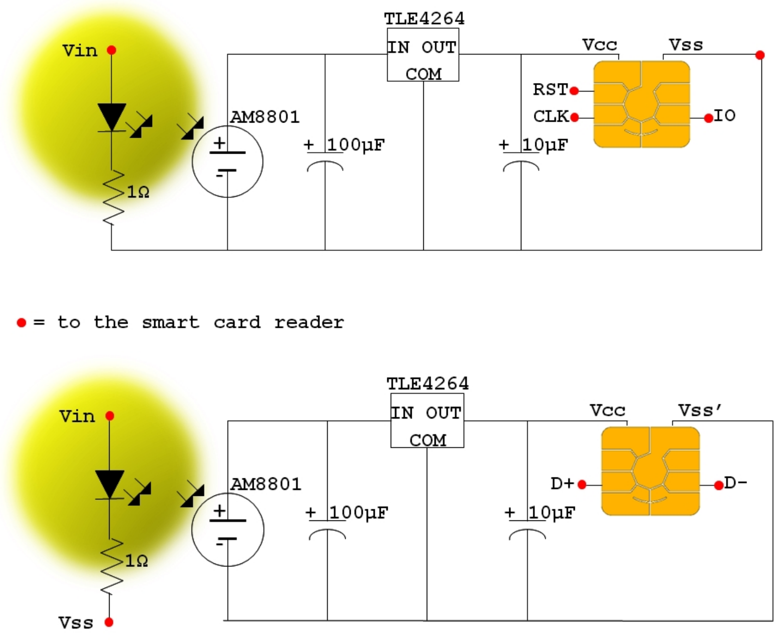


Figure 6.9: Electronic scheme of the countermeasure and card contacts, before and after applying the new solution.

7816-3 standards limits and has an excessive heat dissipation. But, as more than 41.2% of our OLED panel was not covered by the photovoltaic cell, the consumption can be reduced to about 58.7% of the current value if the size of OLED panel fits 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 might be resolved.

It seems that our Osram Orbeos panel is the only commercialized OLED at the moment. The upcoming OLED panels in few next months have better efficiency, *e.g.* 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).

Some other technical facts also should be considered for a commercial application, especially ISO 7816-3 specifications about authorized values of other electrical parameters in a smart card and response time restrictions.

However, our countermeasure can not resolve electromagnetic attacks (EMA). Theoretically, wires might act as antennae and conduct a negligible EMA residue. But in summary, our solution is a countermeasure very close to perfect as it can hide entirely power consumption variations.

Chapter 7

Conclusions and Perspectives

The objective of this thesis was the identification and study of laser fault injection laser threats on the cryptographic systems.

At the first step, we have examined 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 for discovering 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 towards the required models. We succeeded to exclude 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 methods. With the precision obtained in our earlier work, 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. A dozen of these attacks is based on simple scenarios, including an attack at a specific time of algorithm execution, on one of the two aforementioned targets.

Our experiments give a warning for the feasibility of described attacks in the literature by laser. Our tests have demonstrated 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 for 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 cryptosystems security, we participated in few other related researches. We developed a countermeasure against power analysis attacks on smart cards. This countermeasure is composed of an organic light emitting diode (OLED) with a few hundred nanometers thick and a photovoltaic cell having a thickness of several tens of micrometers in order to make consumption of the card independent of the processed data. 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 a part of another research work, we evaluated the resistance of a countermeasure against fault attacks by temperature violations.

Our prospects to continue these researches are:

- **Research on other fault injection mechanisms:** The study of fault injection on the microcontrollers; including the arithmetical logic unit (alu), the bus and the key expansion operations.

- **Study of flash memory vulnerability:** Including research on changing the loaded program using faults and also fault injection on the initial values stored in flash memories.

- **Implementation and validation of countermeasures:** In order to protect the circuits from fault injection and release of sensitive information using faults.

We hope to continue these perspectives for further researches after the current thesis.

Chapter 8

Résumé étendu en français (French Extended Abstract)

Contents

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 symétrique 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 claires 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 claires 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 18ème 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 réalisées 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 principe d'Auguste Kerckhoffs, présentées en 1883 : " Un crypto-système devrait être sûre 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 secrète 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 ensuite en l'exploitation des comportements ou des résultats fautés afin d'en déduire des information 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 développé 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'empêche 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-programmé 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'interrompront 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.

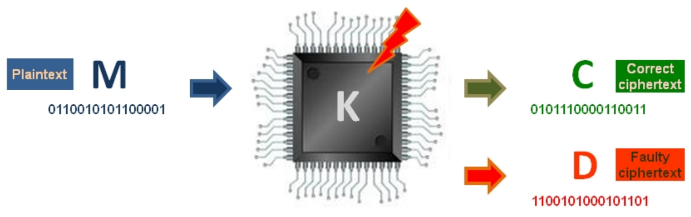


Figure 8.1: Principe de l'analyse différentielle de fautes.

- **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 exploites les conditions sensibles d'apparition de fautes au lieu des résultats fautés 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] [Barenghi 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 $T_1 = \{b_1, b_2, ...b_n\}$ comme un ensemble arbitraire de valeurs initiales pour des bits ciblés. On suppose $T_2 = \{b'_1, b'_2, ...b'_n\}$ 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 :

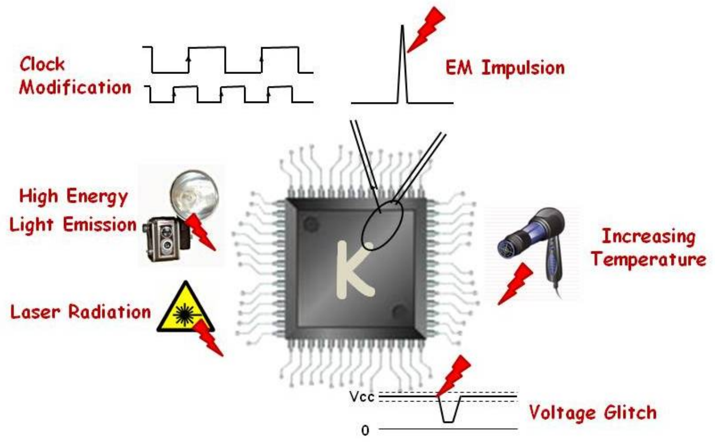


Figure 8.2: Aperçu des principales techniques d'injection de fautes.

- **Basculement ou inversion de bit :** Quand les valeurs de bits ciblés sont changés à leurs valeurs opposées ; soit si et seulement si :

$$\forall i : 0 \leq i \leq n \quad ; \quad b'_i = 1 - b_i \tag{8.1}$$

- **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 :

$$\forall i : 0 \leq i \leq n \quad ; \quad b_i = b'_i = 0 \tag{8.2}$$

  Autrement, le modèle de faute est considéré *collé à 1*, si et seulement si :

$$\forall i : 0 \leq i \leq n \quad ; \quad b_i = b'_i = 1 \tag{8.3}$$

  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 sl aléatoire, si et seulement si :

$$\begin{aligned} \forall i : 0 \leq i \leq n \quad &; \quad b'_i \in \{0, 1\} \\ &\quad\quad\quad\quad \textit{and} \\ \exists! j : 0 \leq j \leq n \quad &; \quad b'_j \neq b_j \end{aligned} \tag{8.4}$$

- **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.

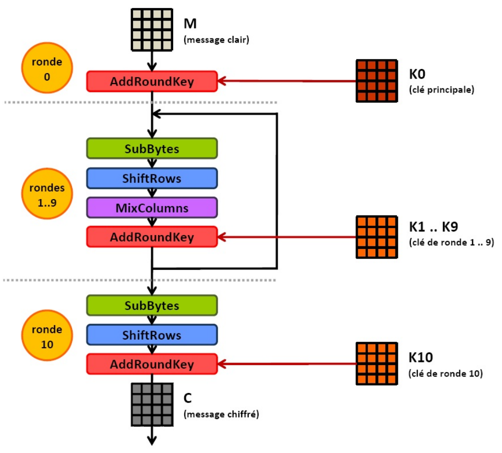


Figure 8.3: Algorithme AES-128.

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 $\varnothing 1\mu$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 pairs 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 YAG[2] 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.

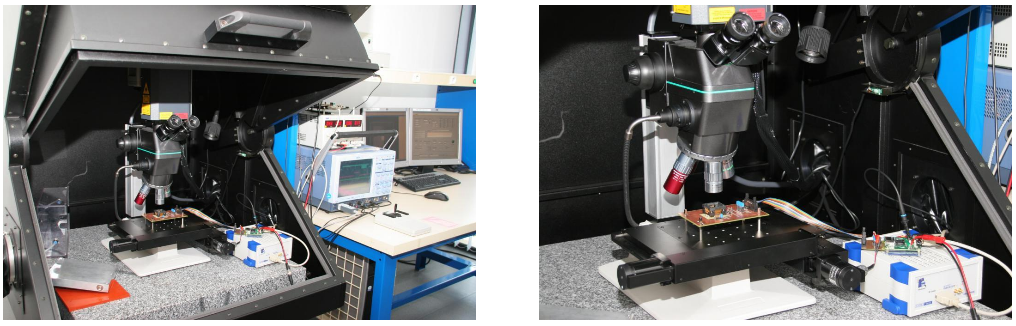


Figure 8.4: Plateforme laser MicroPackS.

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-I[3]. Il s'agit

[2]Grenat d'Yttrium Aluminium ou $Y_3Al_5O_{12}$
[3]**S**ecured **M**ulti-ch**ar**acterisation **T**est **I**nterface

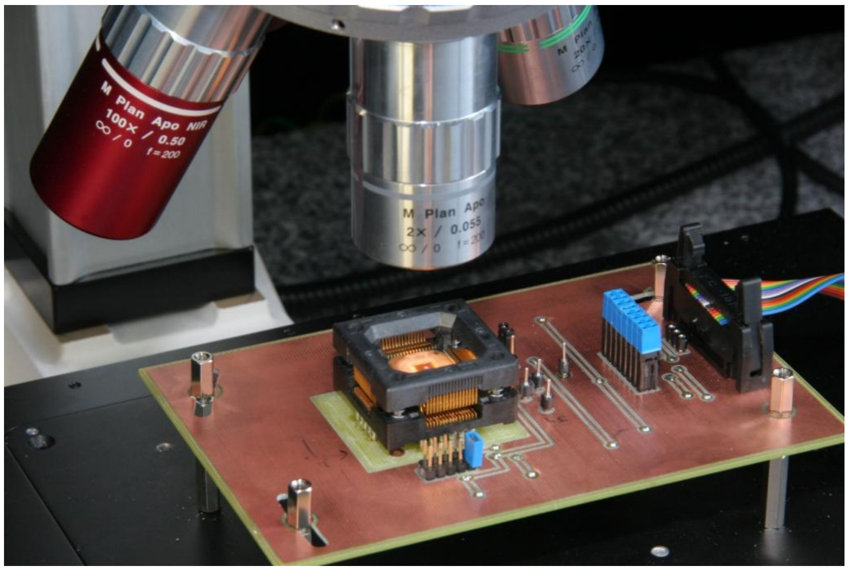


Figure 8.5: Plateforme laser et circuit test.

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 μs, 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 μm 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 au 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 copiés sur la SRAM. Dans notre version, la clef secrète K pour le chiffrement AES est inclus dans le code. Après chaque réinitialisation du circuit, toutes les clefs 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 clefs 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.

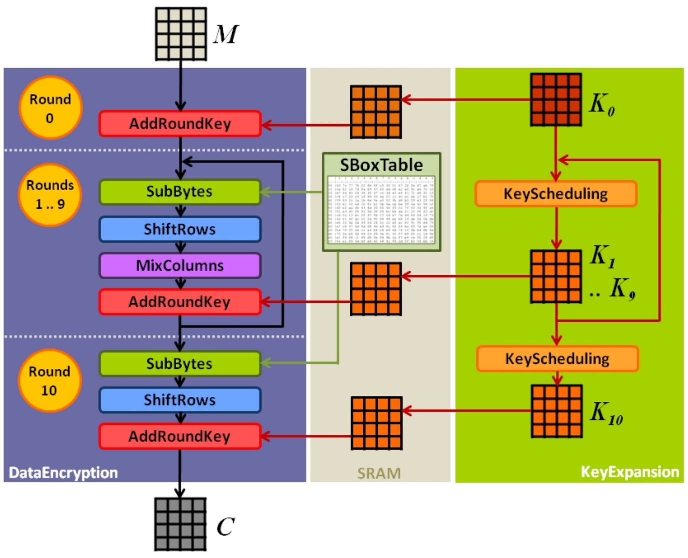


Figure 8.6: Un aperçu de l'implémentation de notre AES.

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 décapsulées, 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.

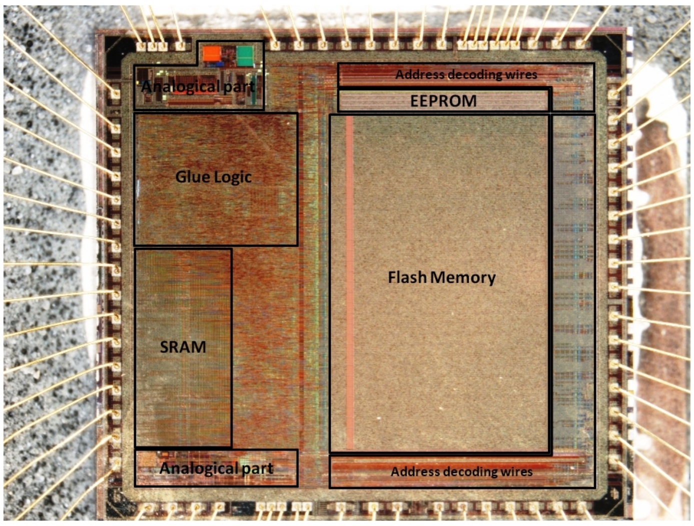


Figure 8.7: Cartographie du microcontrôleur cible.

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 différent 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 $\simeq 44\ \mu$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 μm. 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 classifier 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 $\varnothing 1\mu$m, pour des raisons de diffraction optique. En outre, par suite des progrès technologiques, le nombre de transistors placés sous un faisceau incident de $\varnothing 1\mu$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 $\varnothing 1\mu$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 $\varnothing 1\mu$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

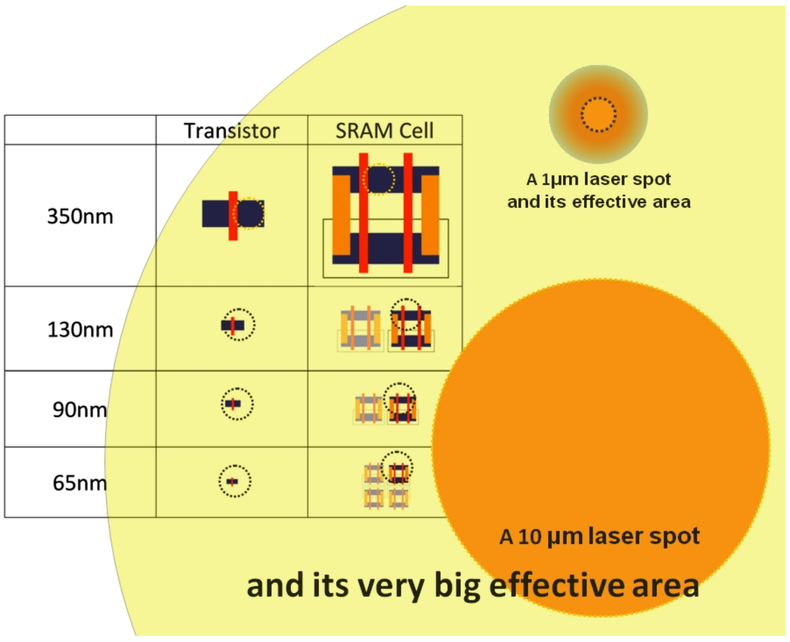


Figure 8.8: Mise en évidence de la taille de faisceaux laser d'1 μm et de 10 μm par rapport au progrès des technologies de fabrication.

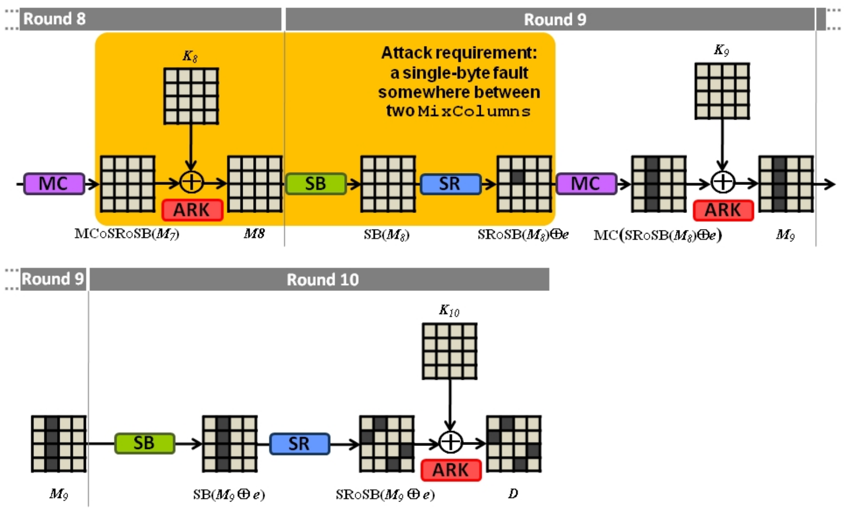


Figure 8.9: Propagation d'une faute mono-octet à l'entrée du `MixColumns` de la ronde 9.

octet spécifique du chiffré temporaire de l'avant-dernière ronde (M_9). La figure 8.10 montre l'exigence de cette attaque.

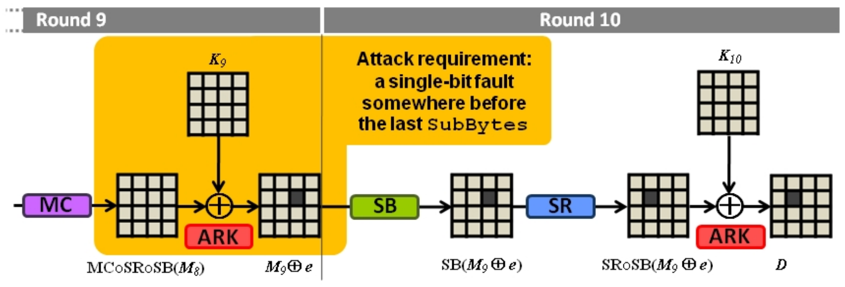


Figure 8.10: Attque DFA Giraud mono-bit.

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 fautés. 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 cryptoprocesseurs asynchrones implémentant l'algorithme de chiffrement DES. Le compteur de rondes utilisé dans ces deux cryptoprocesseurs a une implémentation multi-rail, telle que pour chacune des 16 rondes, une ligne séparée de signal existe. Une $17^{\text{ème}}$ 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 cryptoprocesseurs 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.

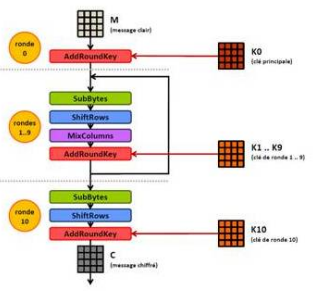
Le travail de J. Park et al. présenté dans la publication [Park 2011], il s'agît d'une attaque en faute par laser sur un AES embarqué dans un circuit Atmega128. Ils ont exploité une attaque de passage de la 1^{re} ronde à la ronde 10. Donc l'exécution fautée comprend les rondes : $R_0 - R_1 - R_{10}$, 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 le détail de l'algorithme de notre AES sur la figure 8.11 :



```
START
    C ← M
    C ← C ⊕ K
    FOR (CR=1 ; CR < Rmax ; CR++)
        {
        C ← SBOX(C)
        C ← SR(C)
        C ← MC(C)
        C ← C ⊕ K_CR
        }
    C ← SBOX(C)
    C ← SR(C)
    C ← C ⊕ K_CR
END
```

Figure 8.11: Implémentation 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 stockées 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 divisions ces différents instants d'exécution de l'algorithme :

1. FOR : Au début du cycle d'une ronde intermédiaire, entre l'affectation ou l'incrémentation 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'affection de la nouvelle valeur à CR

Nous montrons ces différents instants de notre algorithme d'AES sur la figure 8.12.

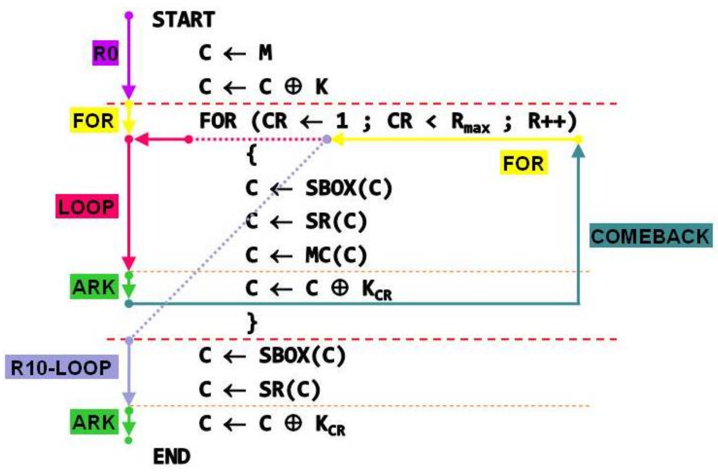


Figure 8.12: Différents moments d'exécution de notre algorithme AES par rapport aux utilisation du compteur de rondes.

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 :

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

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)

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.

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 :

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 COMEBACK	- augmentation ou réduction du nombre de rondes (augmentation ou suppression de quelques rondes ; mais pas de réexécution) - prise en compte de la faute dès le début de la prochaine ronde

- 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 :

  $$\mathrm{R_0..R_5\text{-}R_6\text{-}R_5\text{-}R_6\text{-}R_7..R_{10}}$$

  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 $CR \neq R_{max} - 1 \Rightarrow$ *Réduction de rondes*. Par exemple : si CR=4 et e=2 ; d'après $CR_{\text{fauté}} = CR \oplus e$ =6, l'exécution fautée d'AES sera :

  $$\mathrm{R_0..R_3\text{-}R_6..R_{10}}$$

  Les rondes 4 et 5 seront supprimées et le nombre total des rondes exécutées sera diminué à 8.

- Si $CR \oplus e > CR$ et $CR = R_{max} - 1 \Rightarrow$ 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 :

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

  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^o	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-$R_{\mathrm{m}=10}$ -$R_{\mathrm{f}=11}$	3	Aug.R
II-2	R_{max}	Entre CR=0 et CR=10 FOR accès à la mémoire requis	R_0..R_9-4×$R_{\mathrm{m}=10..13}$ -$R_{\mathrm{f}=14}$	1	Aug.R
II-3	R_{max}	Entre CR=8 LOOP et CR=9 FOR si $e \in\{2, 8\}$	R_0..R_8-$R_{\mathrm{f}=9}$	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 de 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 de 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 par 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.

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 potentiels pour des attaques de modification de rondes. Donc, l'ajout de deux compteur de vérification de rondes, fonctionnant avec des opération

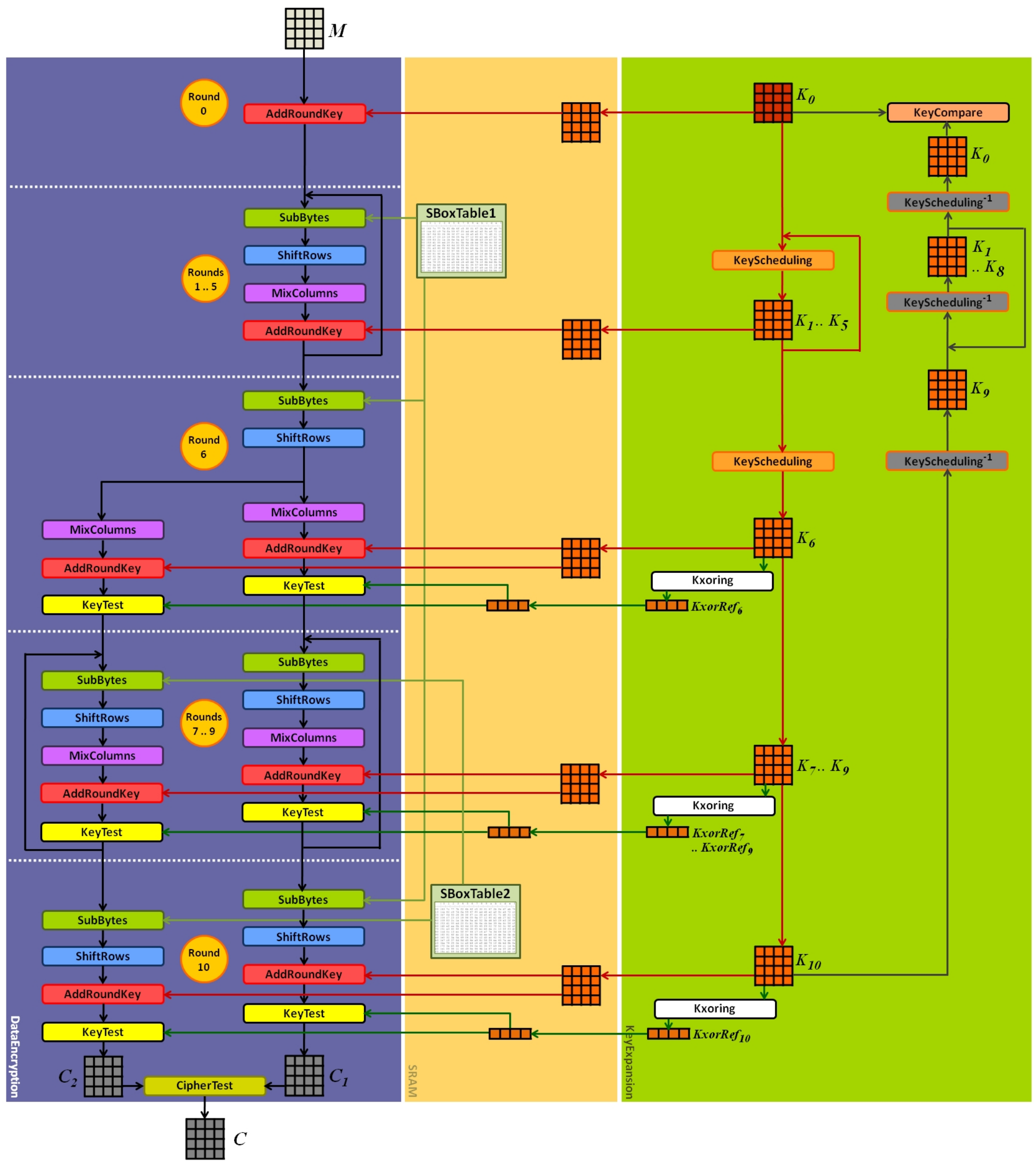


Figure 8.13: Processus parallèles de chiffrement et vérifications des clefs dans notre implémentation d'AES.

de décalage de bit est une solution pour une double vérification et détection de tout changement ou toute altération de rondes.

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

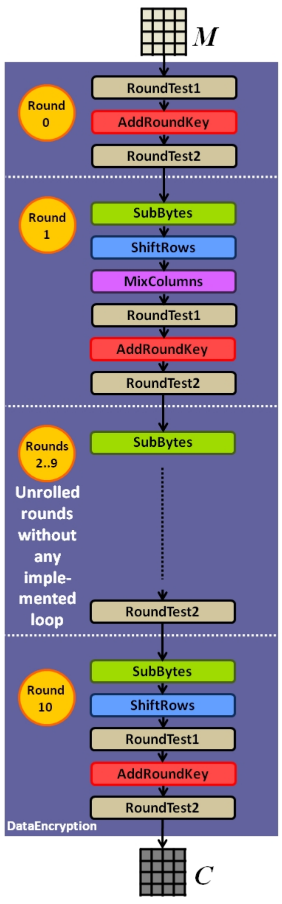


Figure 8.14: Implémentation de l'AES déroulé avec deux vérifications à chaque ronde.

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.

Nous attendons que ces contre-mesures embarquées présentées préservent fortement notre AES des attaques susdites et peuvent être facilement adaptées à d'autres implémentations, afin de les protéger.

Toutefois, elles prennent une partie importante des ressources matérielles de circuits 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.

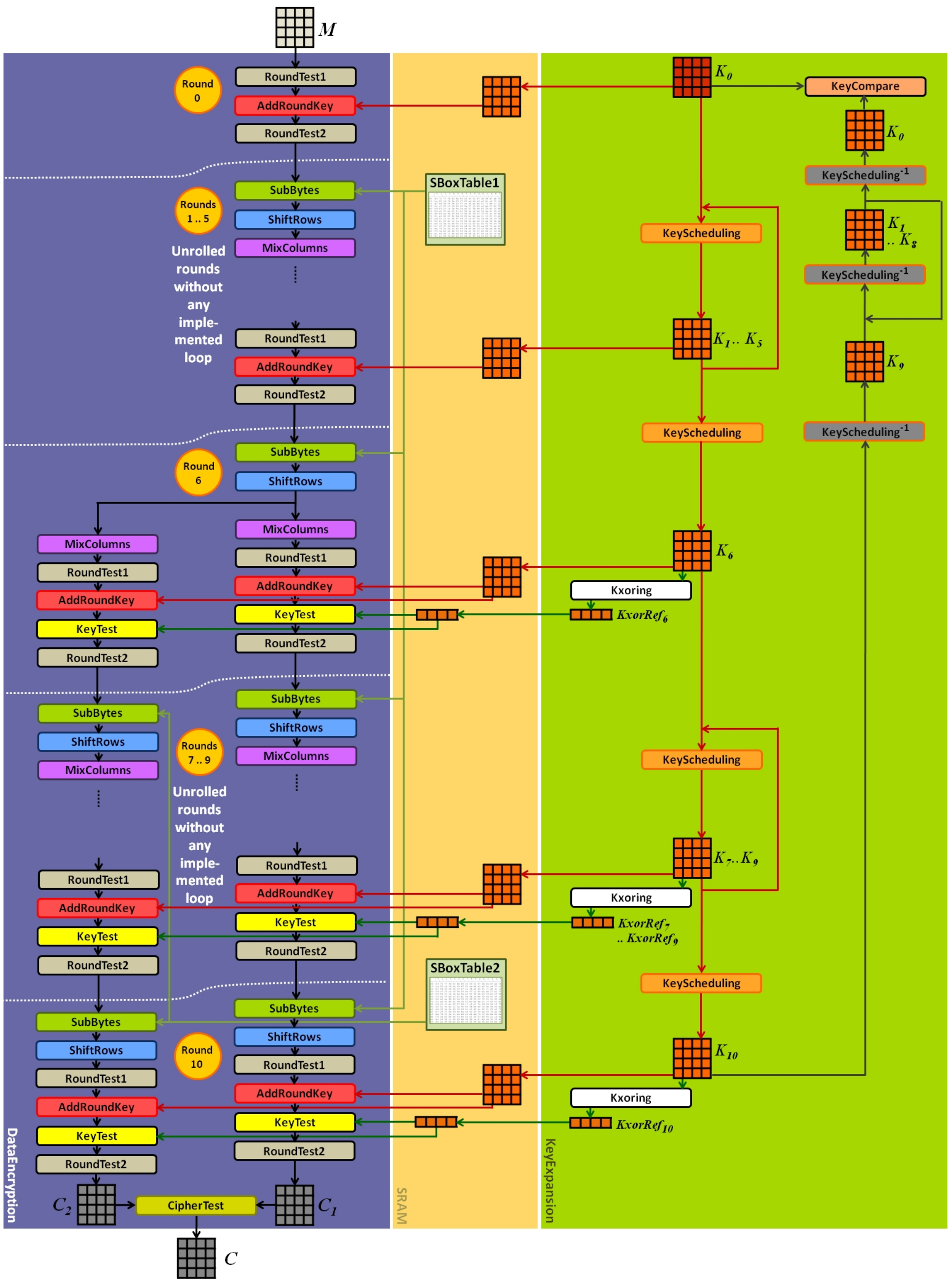


Figure 8.15: Contre-mesure combinée pour des attaques DFA et RMA dans notre implémentation d'AES.

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 très proche du parfait 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 qui fournit 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 d'une source de lumière et une cellule photovoltaïque pourrait être un choix, si elles peuvent satisfaire les restrictions de la taille, de la puissance et des exigences de commutation rapides pour une carte à puce. Entre les sources de lumière, 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 disponible d'un panneau OLED et quelques cellules photovoltaïques de bonnes performances, nous avons examiné leurs efficacités 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 utilisé 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 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 sur $5V$. En plus, nous avons utilisé deux condensateurs électrolytiques de 100 μF et 10 μF, respectivement à l'entrée et la sortie du régulateur.

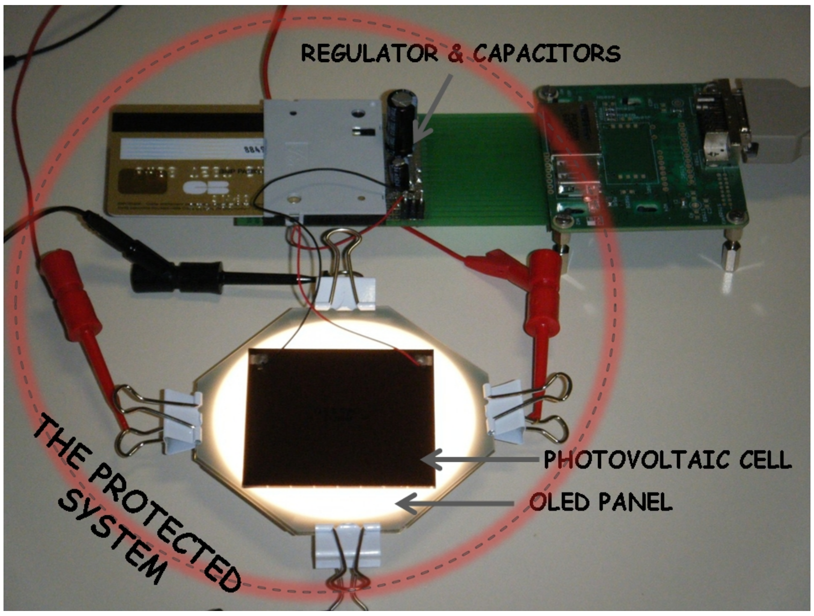


Figure 8.16: Système protégé par la contre-mesure.

La partie inférieure de la figure 8.17 montre le schéma électronique de la contre-mesure des contacts de carte à puce. 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.

Normalement, nous devrions relier toutes les masses comme la partie supérieure de la figure 8.17. Mais, cette configuration pourrait donner la possibilité d'une nouvelle attaque potentielle, par l'analyse de la consommation sur le contact de la 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.

8.7 Conclusions et perspectives

L'objectif de ce travail de thèse était l'identification et l'étude des menaces de l'injection de fautes par laser sur des 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

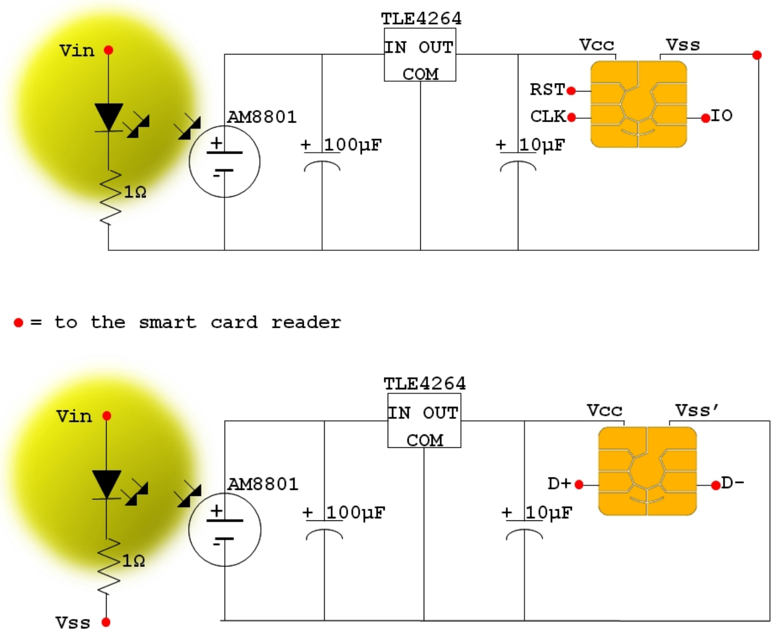


Figure 8.17: Schéma électronique de la contre-mesure des contacts de carte à puce, avant et après la séparation des masses.

laser créait souvent des fautes supplémentaires par rapport aux modèles théoriques de DFA sur AES, requérant l'injection de fautes mono-octets ou mono-bits.

En effet, avec le progrès des technologies de la 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 μm 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 l'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 aussi 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 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. 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 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.

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 A

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 on the round counter (RC) or the reference of total round number (R_{max}). These cases include also the attacks that are not exploitable by light cryptanalysis solutions. The purpose of these tables is an overview of different single-bit attacks. In the case of feasibility of multi-bit fault injection, many other cases must be surveyed.

A.1 Attacks on the Round Counter Value

We show here all the possible algorithm execution when a single-bit fault is injected on 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	$NR\rightarrow 11$ R_0-R_m-R_1..R_{10} m=0	$NR\rightarrow 8$ R_0-R_3..R_{10}	$NR\rightarrow 6$ R_0-R_5..R_{10}	$NR\rightarrow 2$ R_0-R_9..R_{10}	$NR\rightarrow 1$ 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	$NR\rightarrow 11$ R_0-R_m-R_1..R_{10} m=0	$NR\rightarrow 8$ R_0-R_3..R_{10}	$NR\rightarrow 6$ R_0-R_5..R_{10}	$NR\rightarrow 2$ R_0-R_9..R_{10}	$NR\rightarrow 2$ 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	$NR\rightarrow 11$ R_0-R_1-R_1..R_{10}	$NR\rightarrow 8$ R_0-R_1-R_4..R_{10}	$NR\rightarrow 6$ R_0-R_1-R_6..R_{10}	$NR\rightarrow 2$ R_0-R_1-R_{10}	$NR\rightarrow 2$ 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	$NR\rightarrow 9$ R_0-R_1-R_3..R_{10}	$NR\rightarrow 12$ R_0-R_1-R_m-R_1..R_{10} m=0	$NR\rightarrow 6$ R_0-R_1-R_6..R_{10}	$NR\rightarrow 2$ R_0-R_1-R_{10}	$NR\rightarrow 2$ 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	$NR\rightarrow 9$ R_0-R_1-R_3..R_{10}	$NR\rightarrow 12$ R_0-R_1-R_m-R_1..R_{10} m=0	$NR\rightarrow 6$ R_0-R_1-R_6..R_{10}	$NR\rightarrow 3$ R_0-R_1-R_m-R_f m=10 f=11	$NR\rightarrow 3$ 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
	ARK COMEBACK	$NR\rightarrow 9$ R_0..R_2-R_4..R_{10}	$NR\rightarrow 12$ R_0..R_2-R_1..R_{10}	$NR\rightarrow 7$ R_0..R_2-R_6..R_{10}	$NR\rightarrow 3$ R_0..R_2-R_f f=11	$NR\rightarrow 3$ R_0..R_2-R_f f=19	$NR\rightarrow 3$ R_0..R_2-R_f f=35	$NR\rightarrow 3$ R_0..R_2-R_f f=67	$NR\rightarrow 3$ R_0..R_2-R_f f=131

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
$RC = 3$		$RC\to2$	$RC\to1$	$RC\to7$	$RC\to11$	$RC\to19$	$RC\to35$	$RC\to67$	$RC\to131$
	FOR	$NR\to11$ $R_0..R_2$-$R_2..R_{10}$	$NR\to12$ $R_0..R_2$-$R_1..R_{10}$	$NR\to6$ $R_0..R_2$-$R_7..R_{10}$	$NR\to3$ $R_0..R_2$-R_f f=11	$NR\to3$ $R_0..R_2$-R_f f=19	$NR\to3$ $R_0..R_2$-R_f f=35	$NR\to3$ $R_0..R_2$-R_f f=67	$NR\to3$ $R_0..R_2$-R_f f=131
	LOOP	$NR\to11$ $R_0..R_2$-$R_2..R_{10}$	$NR\to12$ $R_0..R_2$-$R_1..R_{10}$	$NR\to6$ $R_0..R_2$-$R_7..R_{10}$	$NR\to4$ $R_0..R_2$-R_m-R_f m=11 f=12	$NR\to4$ $R_0..R_2$-R_m-R_f m=19 f=20	$NR\to4$ $R_0..R_2$-R_m-R_f m=35 f=36	$NR\to4$ $R_0..R_2$-R_m-R_f m=67 f=68	$NR\to4$ $R_0..R_2$-R_m-R_f m=131 f=132
	ARK COMEBACK	$NR\to11$ $R_0..R_3$-$R_3..R_{10}$	$NR\to12$ $R_0..R_3$-$R_2..R_{10}$	$NR\to6$ $R_0..R_3$-$R_8..R_{10}$	$NR\to4$ $R_0..R_3$-R_f f=12	$NR\to4$ $R_0..R_3$-R_f f=20	$NR\to4$ $R_0..R_3$-R_f f=36	$NR\to4$ $R_0..R_3$-R_f f=68	$NR\to4$ $R_0..R_3$-R_f f=132
$RC = 4$		$RC\to5$	$RC\to6$	$RC\to0$	$RC\to12$	$RC\to20$	$RC\to36$	$RC\to68$	$RC\to132$
	FOR	$NR\to9$ $R_0..R_3$-$R_5..R_{10}$	$NR\to8$ $R_0..R_3$-$R_6..R_{10}$	$NR\to14$ $R_0..R_3$-R_m-$R_1..R_{10}$ m=0	$NR\to4$ $R_0..R_3$-R_f f=12	$NR\to4$ $R_0..R_3$-R_f f=20	$NR\to4$ $R_0..R_3$-R_f f=36	$NR\to4$ $R_0..R_3$-R_f f=68	$NR\to4$ $R_0..R_3$-R_f f=132
	LOOP	$NR\to9$ $R_0..R_3$-$R_5..R_{10}$	$NR\to8$ $R_0..R_3$-$R_6..R_{10}$	$NR\to14$ $R_0..R_3$-R_m-$R_1..R_{10}$ m=0	$NR\to5$ $R_0..R_3$-R_m-R_f m=12 f=13	$NR\to5$ $R_0..R_3$-R_m-R_f m=20 f=21	$NR\to5$ $R_0..R_3$-R_m-R_f m=36 f=37	$NR\to5$ $R_0..R_3$-R_m-R_f m=68 f=69	$NR\to5$ $R_0..R_3$-R_m-R_f m=132 f=133
	ARK COMEBACK	$NR\to9$ $R_0..R_4$-$R_6..R_{10}$	$NR\to8$ $R_0..R_4$-$R_7..R_{10}$	$NR\to14$ $R_0..R_4$-$R_1..R_{10}$	$NR\to5$ $R_0..R_4$-R_f f=13	$NR\to5$ $R_0..R_4$-R_f f=21	$NR\to5$ $R_0..R_4$-R_f f=37	$NR\to5$ $R_0..R_4$-R_f f=69	$NR\to5$ $R_0..R_4$-R_f f=133

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
$RC = 5$		$RC\to 4$	$RC\to 7$	$RC\to 1$	$RC\to 13$	$RC\to 21$	$RC\to 37$	$RC\to 69$	$RC\to 133$
	FOR	$NR\to 11$ $R_0..R_4\text{-}R_4..R_{10}$	$NR\to 8$ $R_0..R_4\text{-}R_7..R_{10}$	$NR\to 14$ $R_0..R_4\text{-}R_1..R_{10}$	$NR\to 5$ $R_0..R_4\text{-}R_f$ f=13	$NR\to 5$ $R_0..R_4\text{-}R_f$ f=21	$NR\to 5$ $R_0..R_4\text{-}R_f$ f=37	$NR\to 5$ $R_0..R_4\text{-}R_f$ f=69	$NR\to 5$ $R_0..R_4\text{-}R_f$ f=133
	LOOP	$NR\to 11$ $R_0..R_4\text{-}R_4..R_{10}$	$NR\to 8$ $R_0..R_4\text{-}R_7..R_{10}$	$NR\to 14$ $R_0..R_4\text{-}R_1..R_{10}$	$NR\to 6$ $R_0..R_4\text{-}R_m\text{-}R_f$ m=13 f=14	$NR\to 6$ $R_0..R_4\text{-}R_m\text{-}R_f$ m=21 f=22	$NR\to 6$ $R_0..R_4\text{-}R_m\text{-}R_f$ m=37 f=38	$NR\to 6$ $R_0..R_4\text{-}R_m\text{-}R_f$ m=69 f=70	$NR\to 6$ $R_0..R_4\text{-}R_m\text{-}R_f$ m=133 f=134
	ARK COMEBACK	$NR\to 11$ $R_0..R_5\text{-}R_5..R_{10}$	$NR\to 8$ $R_0..R_5\text{-}R_8..R_{10}$	$NR\to 14$ $R_0..R_5\text{-}R_2..R_{10}$	$NR\to 6$ $R_0..R_5\text{-}R_f$ f=14	$NR\to 6$ $R_0..R_5\text{-}R_f$ f=22	$NR\to 6$ $R_0..R_5\text{-}R_f$ f=38	$NR\to 6$ $R_0..R_5\text{-}R_f$ f=70	$NR\to 6$ $R_0..R_5\text{-}R_f$ f=134
$RC = 6$		$RC\to 7$	$RC\to 4$	$RC\to 2$	$RC\to 14$	$RC\to 22$	$RC\to 38$	$RC\to 70$	$RC\to 134$
	FOR	$NR\to 9$ $R_0..R_5\text{-}R_7..R_{10}$	$NR\to 12$ $R_0..R_5\text{-}R_4..R_{10}$	$NR\to 14$ $R_0..R_5\text{-}R_2..R_{10}$	$NR\to 6$ $R_0..R_5\text{-}R_f$ f=14	$NR\to 6$ $R_0..R_5\text{-}R_f$ f=22	$NR\to 6$ $R_0..R_5\text{-}R_f$ f=38	$NR\to 6$ $R_0..R_5\text{-}R_f$ f=70	$NR\to 6$ $R_0..R_5\text{-}R_f$ f=134
	LOOP	$NR\to 9$ $R_0..R_5\text{-}R_7..R_{10}$	$NR\to 12$ $R_0..R_5\text{-}R_4..R_{10}$	$NR\to 14$ $R_0..R_5\text{-}R_2..R_{10}$	$NR\to 7$ $R_0..R_5\text{-}R_m\text{-}R_f$ m=14 f=15	$NR\to 7$ $R_0..R_5\text{-}R_m\text{-}R_f$ m=22 f=23	$NR\to 7$ $R_0..R_5\text{-}R_m\text{-}R_f$ m=38 f=39	$NR\to 7$ $R_0..R_5\text{-}R_m\text{-}R_f$ m=70 f=71	$NR\to 7$ $R_0..R_5\text{-}R_m\text{-}R_f$ m=134 f=135
	ARK COMEBACK	$NR\to 9$ $R_0..R_6\text{-}R_8..R_{10}$	$NR\to 12$ $R_0..R_6\text{-}R_5..R_{10}$	$NR\to 14$ $R_0..R_6\text{-}R_3..R_{10}$	$NR\to 7$ $R_0..R_6\text{-}R_f$ f=15	$NR\to 7$ $R_0..R_6\text{-}R_f$ f=23	$NR\to 7$ $R_0..R_6\text{-}R_f$ f=39	$NR\to 7$ $R_0..R_6\text{-}R_f$ f=71	$NR\to 7$ $R_0..R_6\text{-}R_f$ 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
$RC = 7$		$RC\to6$	$RC\to5$	$RC\to3$	$RC\to15$	$RC\to23$	$RC\to39$	$RC\to71$	$RC\to135$
	FOR	$NR\to11$ R_0..R_6-R_6..R_{10}	$NR\to12$ R_0..R_6-R_5..R_{10}	$NR\to14$ R_0..R_6-R_3..R_{10}	$NR\to7$ R_0..R_6-R_f f=15	$NR\to7$ R_0..R_6-R_f f=23	$NR\to7$ R_0..R_6-R_f f=39	$NR\to7$ R_0..R_6-R_f f=71	$NR\to7$ R_0..R_6-R_f f=135
	LOOP	$NR\to11$ R_0..R_6-R_6..R_{10}	$NR\to12$ R_0..R_6-R_5..R_{10}	$NR\to14$ R_0..R_6-R_3..R_{10}	$NR\to8$ R_0..R_6-R_m-R_f m=15 f=16	$NR\to8$ R_0..R_6-R_m-R_f m=23 f=24	$NR\to8$ R_0..R_6-R_m-R_f m=39 f=40	$NR\to8$ R_0..R_6-R_m-R_f m=71 f=72	$NR\to8$ R_0..R_6-R_m-R_f m=135 f=136
	ARK COMEBACK	$NR\to11$ R_0..R_7-R_7..R_{10}	$NR\to12$ R_0..R_7-R_6..R_{10}	$NR\to14$ R_0..R_7-R_4..R_{10}	$NR\to8$ R_0..R_7-R_f f=16	$NR\to8$ R_0..R_7-R_f f=24	$NR\to8$ R_0..R_7-R_f f=40	$NR\to8$ R_0..R_7-R_f f=72	$NR\to8$ R_0..R_7-R_f f=136
$RC = 8$		$RC\to9$	$RC\to10$	$RC\to12$	$RC\to0$	$RC\to24$	$RC\to40$	$RC\to72$	$RC\to136$
	FOR	$NR\to9$ R_0..R_7-R_9..R_{10}	$NR\to8$ R_0..R_7-R_{10}	$NR\to8$ R_0..R_7-R_f f=12	$NR\to18$ R_0..R_7-R_m-R_1..R_{10} m=0	$NR\to8$ R_0..R_7-R_f f=24	$NR\to8$ R_0..R_7-R_f f=40	$NR\to8$ R_0..R_7-R_f f=72	$NR\to8$ R_0..R_7-R_f f=136
	LOOP	$NR\to9$ R_0..R_7-R_9..R_{10}	$NR\to9$ R_0..R_7-R_m-R_f m=10 f=11	$NR\to9$ R_0..R_7-R_m-R_f m=12 f=13	$NR\to18$ R_0..R_7-R_m-R_1-R_{10} m=0	$NR\to9$ R_0..R_7-R_m-R_f m=24 f=25	$NR\to9$ R_0..R_7-R_m-R_f m=40 f=41	$NR\to9$ R_0..R_7-R_m-R_f m=72 f=73	$NR\to9$ R_0..R_7-R_m-R_f m=136 f=137
	ARK COMEBACK	$NR\to9$ R_0..R_8-R_{10}	$NR\to9$ R_0..R_8-R_f f=11	$NR\to9$ R_0..R_8-R_f f=13	$NR\to18$ R_0..R_8-R_1..R_{10}	$NR\to9$ R_0..R_8-R_f f=25	$NR\to9$ R_0..R_8-R_f f=41	$NR\to9$ R_0..R_8-R_f f=73	$NR\to9$ R_0..R_8-R_f f=137

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
$RC = 9$		$RC\to 8$	$RC\to 11$	$RC\to 13$	$RC\to 1$	$RC\to 25$	$RC\to 41$	$RC\to 73$	$RC\to 137$
	FOR	$NR\to 11$ $R_0..R_8\text{-}R_8..R_{10}$	$NR\to 9$ $R_0..R_8\text{-}R_f$ f=11	$NR\to 9$ $R_0..R_8\text{-}R_f$ f=13	$NR\to 18$ $R_0..R_8\text{-}R_1..R_{10}$	$NR\to 9$ $R_0..R_8\text{-}R_f$ f=25	$NR\to 9$ $R_0..R_8\text{-}R_f$ f=41	$NR\to 9$ $R_0..R_8\text{-}R_f$ f=73	$NR\to 9$ $R_0..R_8\text{-}R_f$ f=137
	LOOP	$NR\to 11$ $R_0..R_8\text{-}R_8..R_{10}$	$NR\to 10$ $R_0..R_8\text{-}R_m\text{-}R_f$ m=11 f=12	$NR\to 10$ $R_0..R_8\text{-}R_m\text{-}R_f$ m=13 f=14	$NR\to 18$ $R_0..R_8\text{-}R_1..R_{10}$	$NR\to 10$ $R_0..R_8\text{-}R_m\text{-}R_f$ m=25 f=26	$NR\to 10$ $R_0..R_8\text{-}R_m\text{-}R_f$ m=41 f=42	$NR\to 10$ $R_0..R_8\text{-}R_m\text{-}R_f$ m=73 f=74	$NR\to 10$ $R_0..R_8\text{-}R_m\text{-}R_f$ m=137 f=138
	ARK COMEBACK	$NR\to 11$ $R_0..R_9\text{-}R_9..R_{10}$	$NR\to 10$ $R_0..R_9\text{-}R_f$ f=12	$NR\to 10$ $R_0..R_9\text{-}R_f$ f=14	$NR\to 18$ $R_0..R_9\text{-}R_2..R_{10}$	$NR\to 10$ $R_0..R_9\text{-}R_f$ f=26	$NR\to 10$ $R_0..R_9\text{-}R_f$ f=42	$NR\to 10$ $R_0..R_9\text{-}R_f$ f=74	$NR\to 10$ $R_0..R_9\text{-}R_f$ f=138
$RC = 10$		$RC\to 11$	$RC\to 8$	$RC\to 14$	$RC\to 2$	$RC\to 26$	$RC\to 42$	$RC\to 74$	$RC\to 138$
	FOR	$NR\to 10$ $R_0..R_9\text{-}R_f$ f=11	$NR\to 12$ $R_0..R_9\text{-}R_8..R_{10}$	$NR\to 10$ $R_0..R_9\text{-}R_f$ f=14	$NR\to 18$ $R_0..R_9\text{-}R_2..R_{10}$	$NR\to 10$ $R_0..R_9\text{-}R_f$ f=26	$NR\to 10$ $R_0..R_9\text{-}R_f$ f=42	$NR\to 10$ $R_0..R_9\text{-}R_f$ f=74	$NR\to 10$ $R_0..R_9\text{-}R_f$ f=138
	LOOP	$NR\to 10$ $R_0..R_9\text{-}R_f$ f=11	$NR\to 10$ $R_0..R_9\text{-}R_f$ f=8	$NR\to 10$ $R_0..R_9\text{-}R_f$ f=14	$NR\to 10$ $R_0..R_9\text{-}R_f$ f=2	$NR\to 10$ $R_0..R_9\text{-}R_f$ f=26	$NR\to 10$ $R_0..R_9\text{-}R_f$ f=42	$NR\to 10$ $R_0..R_9\text{-}R_f$ f=74	$NR\to 10$ $R_0..R_9\text{-}R_f$ f=138

A.2 Attacks on the Round Number Reference

We show here all the possible algorithm execution 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: All the single-bit attacks of scenario II.

Round counter	Attack moment	Model of injected fault							
		0x01	0x02	0x04	0x08	0x10	0x20	0x40	0x80
		$R_{max}\rightarrow 11$	$R_{max}\rightarrow 8$	$R_{max}\rightarrow 14$	$R_{max}\rightarrow 2$	$R_{max}\rightarrow 26$	$R_{max}\rightarrow 42$	$R_{max}\rightarrow 74$	$R_{max}\rightarrow 138$
$RC = 1$									
	FOR LOOP ARK COMEBACK	$NR\rightarrow 11$ $R_0..R_9\text{-}R_m\text{-}R_f$ m=10 f=11	$NR\rightarrow 8$ $R_0..R_7\text{-}R_f$ f=8	$NR\rightarrow 14$ $R_0..R_9\text{-}4\times R_m\text{-}R_f$ m={10, 11, .. 13} f=14	$NR\rightarrow 2$ $R_0..R_1\text{-}R_f$ f=2	$NR\rightarrow 26$ $R_0..R_9\text{-}16\times R_m\text{-}R_f$ m={10, 11, .. 25} f=26	$NR\rightarrow 42$ $R_0..R_9\text{-}32\times R_m\text{-}R_f$ m={10, 11, .. 41} f=42	$NR\rightarrow 74$ $R_0..R_9\text{-}64\times R_m\text{-}R_f$ m={10, 11, .. 73} f=74	$NR\rightarrow 138$ $R_0..R_9\text{-}128\times R_m\text{-}R_f$ m={10, 11, .. 137} f=138
$RC = 2$									
	FOR	$NR\rightarrow 11$ $R_0..R_9\text{-}R_m\text{-}R_f$ m=10 f=11	$NR\rightarrow 8$ $R_0..R_7\text{-}R_f$ f=8	$NR\rightarrow 14$ $R_0..R_9\text{-}4\times R_m\text{-}R_f$ m={10, 11, .. 13} f=14	$NR\rightarrow 2$ $R_0..R_1\text{-}R_f$ f=2	$NR\rightarrow 26$ $R_0..R_9\text{-}16\times R_m\text{-}R_f$ m={10, 11, .. 25} f=26	$NR\rightarrow 42$ $R_0..R_9\text{-}32\times R_m\text{-}R_f$ m={10, 11, .. 41} f=42	$NR\rightarrow 74$ $R_0..R_9\text{-}64\times R_m\text{-}R_f$ m={10, 11, .. 73} f=74	$NR\rightarrow 138$ $R_0..R_9\text{-}128\times R_m\text{-}R_f$ m={10, 11, .. 137} f=138
	LOOP ARK COMEBACK	$NR\rightarrow 11$ $R_0..R_9\text{-}R_m\text{-}R_f$ m=10 f=11	$NR\rightarrow 8$ $R_0..R_7\text{-}R_f$ f=8	$NR\rightarrow 14$ $R_0..R_9\text{-}4\times R_m\text{-}R_f$ m={10, 11, .. 13} f=14	$NR\rightarrow 3$ $R_0..R_2\text{-}R_f$ f=3	$NR\rightarrow 26$ $R_0..R_9\text{-}16\times R_m\text{-}R_f$ m={10, 11, .. 25} f=26	$NR\rightarrow 42$ $R_0..R_9\text{-}32\times R_m\text{-}R_f$ m={10, 11, .. 41} f=42	$NR\rightarrow 74$ $R_0..R_9\text{-}64\times R_m\text{-}R_f$ m={10, 11, .. 73} f=74	$NR\rightarrow 138$ $R_0..R_9\text{-}128\times R_m\text{-}R_f$ m={10, 11, .. 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
		R_{max}→11	R_{max}→8	R_{max}→14	R_{max}→2	R_{max}→26	R_{max}→42	R_{max}→74	R_{max}→138
$RC = 3$									
	FOR	NR→11 R_0..R_9-R_m-R_f m=10 f=11	NR→8 R_0..R_7-R_f f=8	NR→14 R_0..R_9-4×R_m-R_f m={10, 11, .. 13} f=14	NR→3 R_0..R_2-R_f f=3	NR→26 R_0..R_9-16×R_m-R_f m={10, 11, .. 25} f=26	NR→42 R_0..R_9-32×R_m-R_f m={10, 11, .. 41} f=42	NR→74 R_0..R_9-64×R_m-R_f m={10, 11, .. 73} f=74	NR→138 R_0..R_9-128×R_m-R_f m={10, 11, .. 137} f=138
	LOOP ARK COMEBACK	NR→11 R_0..R_9-R_m-R_f m=10 f=11	NR→8 R_0..R_7-R_f f=8	NR→14 R_0..R_9-4×R_m-R_f m={10, 11, .. 13} f=14	NR→4 R_0..R_3-R_f f=4	NR→26 R_0..R_9-16×R_m-R_f m={10, 11, .. 25} f=26	NR→42 R_0..R_9-32×R_m-R_f m={10, 11, .. 41} f=42	NR→74 R_0..R_9-64×R_m-R_f m={10, 11, .. 73} f=74	NR→138 R_0..R_9-128×R_m-R_f m={10, 11, .. 137} f=138
$RC = 4$									
	FOR	NR→11 R_0..R_9-R_m-R_f m=10 f=11	NR→8 R_0..R_7-R_f f=8	NR→14 R_0..R_9-4×R_m-R_f m={10, 11, .. 13} f=14	NR→4 R_0..R_3-R_f f=4	NR→26 R_0..R_9-16×R_m-R_f m={10, 11, .. 25} f=26	NR→42 R_0..R_9-32×R_m-R_f m={10, 11, .. 41} f=42	NR→74 R_0..R_9-64×R_m-R_f m={10, 11, .. 73} f=74	NR→138 R_0..R_9-128×R_m-R_f m={10, 11, .. 137} f=138
	LOOP ARK COMEBACK	NR→11 R_0..R_9-R_m-R_f m=10 f=11	NR→8 R_0..R_7-R_f f=8	NR→14 R_0..R_9-4×R_m-R_f m={10, 11, .. 13} f=14	NR→5 R_0..R_4-R_f f=5	NR→26 R_0..R_9-16×R_m-R_f m={10, 11, .. 25} f=26	NR→42 R_0..R_9-32×R_m-R_f m={10, 11, .. 41} f=42	NR→74 R_0..R_9-64×R_m-R_f m={10, 11, .. 73} f=74	NR→138 R_0..R_9-128×R_m-R_f m={10, 11, .. 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
		$R_{max}\to 11$	$R_{max}\to 8$	$R_{max}\to 14$	$R_{max}\to 2$	$R_{max}\to 26$	$R_{max}\to 42$	$R_{max}\to 74$	$R_{max}\to 138$
$RC = 5$									
	FOR	$NR\to 11$ $R_0..R_9$-R_m-R_f m=10 f=11	$NR\to 8$ $R_0..R_7$-R_f f=8	$NR\to 14$ $R_0..R_9$-$4\times R_m$-R_f m={10, 11, .. 13} f=14	$NR\to 5$ $R_0..R_4$-R_f f=5	$NR\to 26$ $R_0..R_9$-$16\times R_m$-R_f m={10, 11, .. 25} f=26	$NR\to 42$ $R_0..R_9$-$32\times R_m$-R_f m={10, 11, .. 41} f=42	$NR\to 74$ $R_0..R_9$-$64\times R_m$-R_f m={10, 11, .. 73} f=74	$NR\to 138$ $R_0..R_9$-$128\times R_m$-R_f m={10, 11, .. 137} f=138
	LOOP ARK COMEBACK	$NR\to 11$ $R_0..R_9$-R_m-R_f m=10 f=11	$NR\to 8$ $R_0..R_7$-R_f f=8	$NR\to 14$ $R_0..R_9$-$4\times R_m$-R_f m={10, 11, .. 13} f=14	$NR\to 6$ $R_0..R_5$-R_f f=6	$NR\to 26$ $R_0..R_9$-$16\times R_m$-R_f m={10, 11, .. 25} f=26	$NR\to 42$ $R_0..R_9$-$32\times R_m$-R_f m={10, 11, .. 41} f=42	$NR\to 74$ $R_0..R_9$-$64\times R_m$-R_f m={10, 11, .. 73} f=74	$NR\to 138$ $R_0..R_9$-$128\times R_m$-R_f m={10, 11, .. 137} f=138
$RC = 6$									
	FOR	$NR\to 11$ $R_0..R_9$-R_m-R_f m=10 f=11	$NR\to 8$ $R_0..R_7$-R_f f=8	$NR\to 14$ $R_0..R_9$-$4\times R_m$-R_f m={10, 11, .. 13} f=14	$NR\to 6$ $R_0..R_5$-R_f f=6	$NR\to 26$ $R_0..R_9$-$16\times R_m$-R_f m={10, 11, .. 25} f=26	$NR\to 42$ $R_0..R_9$-$32\times R_m$-R_f m={10, 11, .. 41} f=42	$NR\to 74$ $R_0..R_9$-$64\times R_m$-R_f m={10, 11, .. 73} f=74	$NR\to 138$ $R_0..R_9$-$128\times R_m$-R_f m={10, 11, .. 137} f=138
	LOOP ARK COMEBACK	$NR\to 11$ $R_0..R_9$-R_m-R_f m=10 f=11	$NR\to 8$ $R_0..R_7$-R_f f=8	$NR\to 14$ $R_0..R_9$-$4\times R_m$-R_f m={10, 11, .. 13} f=14	$NR\to 7$ $R_0..R_6$-R_f f=7	$NR\to 26$ $R_0..R_9$-$16\times R_m$-R_f m={10, 11, .. 25} f=26	$NR\to 42$ $R_0..R_9$-$32\times R_m$-R_f m={10, 11, .. 41} f=42	$NR\to 74$ $R_0..R_9$-$64\times R_m$-R_f m={10, 11, .. 73} f=74	$NR\to 138$ $R_0..R_9$-$128\times R_m$-R_f m={10, 11, .. 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
		$R_{max}\to11$	$R_{max}\to8$	$R_{max}\to14$	$R_{max}\to2$	$R_{max}\to26$	$R_{max}\to42$	$R_{max}\to74$	$R_{max}\to138$
$RC = 7$									
	FOR	$NR\to11$ $R_0..R_9\text{-}R_m\text{-}R_f$ m=10 f=11	$NR\to8$ $R_0..R_7\text{-}R_f$ f=8	$NR\to14$ $R_0..R_9\text{-}4\times R_m\text{-}R_f$ m={10, 11, .. 13} f=14	$NR\to7$ $R_0..R_6\text{-}R_f$ f=7	$NR\to26$ $R_0..R_9\text{-}16\times R_m\text{-}R_f$ m={10, 11, .. 25} f=26	$NR\to42$ $R_0..R_9\text{-}32\times R_m\text{-}R_f$ m={10, 11, .. 41} f=42	$NR\to74$ $R_0..R_9\text{-}64\times R_m\text{-}R_f$ m={10, 11, .. 73} f=74	$NR\to138$ $R_0..R_9\text{-}128\times R_m\text{-}R_f$ m={10, 11, .. 137} f=138
	LOOP ARK COMEBACK	$NR\to11$ $R_0..R_9\text{-}R_m\text{-}R_f$ m=10 f=11	$NR\to8$ $R_0..R_7\text{-}R_f$ f=8	$NR\to14$ $R_0..R_9\text{-}4\times R_m\text{-}R_f$ m={10, 11, .. 13} f=14	$NR\to8$ $R_0..R_7\text{-}R_f$ f=8	$NR\to26$ $R_0..R_9\text{-}16\times R_m\text{-}R_f$ m={10, 11, .. 25} f=26	$NR\to42$ $R_0..R_9\text{-}32\times R_m\text{-}R_f$ m={10, 11, .. 41} f=42	$NR\to74$ $R_0..R_9\text{-}64\times R_m\text{-}R_f$ m={10, 11, .. 73} f=74	$NR\to138$ $R_0..R_9\text{-}128\times R_m\text{-}R_f$ m={10, 11, .. 137} f=138
$RC = 8$									
	FOR	$NR\to11$ $R_0..R_9\text{-}R_m\text{-}R_f$ m=10 f=11	$NR\to8$ $R_0..R_7\text{-}R_f$ f=8	$NR\to14$ $R_0..R_9\text{-}4\times R_m\text{-}R_f$ m={10, 11, .. 13} f=14	$NR\to8$ $R_0..R_7\text{-}R_f$ f=8	$NR\to26$ $R_0..R_9\text{-}16\times R_m\text{-}R_f$ m={10, 11, .. 25} f=26	$NR\to42$ $R_0..R_9\text{-}32\times R_m\text{-}R_f$ m={10, 11, .. 41} f=42	$NR\to74$ $R_0..R_9\text{-}64\times R_m\text{-}R_f$ m={10, 11, .. 73} f=74	$NR\to138$ $R_0..R_9\text{-}128\times R_m\text{-}R_f$ m={10, 11, .. 137} f=138
	LOOP ARK COMEBACK	$NR\to11$ $R_0..R_9\text{-}R_m\text{-}R_f$ m=10 f=11	$NR\to9$ $R_0..R_8\text{-}R_f$ f=9	$NR\to14$ $R_0..R_9\text{-}4\times R_m\text{-}R_f$ m={10, 11, .. 13} f=14	$NR\to9$ $R_0..R_8\text{-}R_f$ f=9	$NR\to26$ $R_0..R_9\text{-}16\times R_m\text{-}R_f$ m={10, 11, .. 25} f=26	$NR\to42$ $R_0..R_9\text{-}32\times R_m\text{-}R_f$ m={10, 11, .. 41} f=42	$NR\to74$ $R_0..R_9\text{-}64\times R_m\text{-}R_f$ m={10, 11, .. 73} f=74	$NR\to138$ $R_0..R_9\text{-}128\times R_m\text{-}R_f$ m={10, 11, .. 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
		$R_{max}\to11$	$R_{max}\to8$	$R_{max}\to14$	$R_{max}\to2$	$R_{max}\to26$	$R_{max}\to42$	$R_{max}\to74$	$R_{max}\to138$
$RC = 9$									
	FOR	$NR\to11$ $R_0..R_9$-R_m-R_f m=10 f=11	$NR\to9$ $R_0..R_8$-R_f f=9	$NR\to14$ $R_0..R_9$-$4\times R_m$-R_f m={10, 11, .. 13} f=14	$NR\to9$ $R_0..R_8$-R_f f=9	$NR\to26$ $R_0..R_9$-$16\times R_m$-R_f m={10, 11, .. 25} f=26	$NR\to42$ $R_0..R_9$-$32\times R_m$-R_f m={10, 11, .. 41} f=42	$NR\to74$ $R_0..R_9$-$64\times R_m$-R_f m={10, 11, .. 73} f=74	$NR\to138$ $R_0..R_9$-$128\times R_m$-R_f m={10, 11, .. 137} f=138
	LOOP ARK COMEBACK	$NR\to11$ $R_0..R_9$-R_m-R_f m=10 f=11	$NR\to10$ $R_0..R_{10}$ Without effect	$NR\to14$ $R_0..R_9$-$4\times R_m$-R_f m={10, 11, .. 13} f=14	$NR\to10$ $R_0..R_{10}$ Without effect	$NR\to26$ $R_0..R_9$-$16\times R_m$-R_f m={10, 11, .. 25} f=26	$NR\to42$ $R_0..R_9$-$32\times R_m$-R_f m={10, 11, .. 41} f=42	$NR\to74$ $R_0..R_9$-$64\times R_m$-R_f m={10, 11, .. 73} f=74	$NR\to138$ $R_0..R_9$-$128\times R_m$-R_f m={10, 11, .. 137} f=138
$RC = 10$									
	FOR	$NR\to11$ $R_0..R_9$-R_m-R_f m=10 f=11	$NR\to10$ $R_0..R_{10}$ Without effect	$NR\to14$ $R_0..R_9$-$4\times R_m$-R_f m={10, 11, .. 13} f=14	$NR\to10$ $R_0..R_{10}$ Without effect	$NR\to26$ $R_0..R_9$-$16\times R_m$-R_f m={10, 11, .. 25} f=26	$NR\to42$ $R_0..R_9$-$32\times R_m$-R_f m={10, 11, .. 41} f=42	$NR\to74$ $R_0..R_9$-$64\times R_m$-R_f m={10, 11, .. 73} f=74	$NR\to138$ $R_0..R_9$-$128\times R_m$-R_f m={10, 11, .. 137} f=138
	LOOP	$NR\to10$ $R_0..R_{10}$ Without effect	$NR\to10$ $R_0..R_{10}$ Without effect	$NR\to10$ $R_0..R_{10}$ Without effect	$NR\to10$ $R_0..R_{10}$ Without effect	$NR\to10$ $R_0..R_{10}$ Without effect	$NR\to10$ $R_0..R_{10}$ Without effect	$NR\to10$ $R_0..R_{10}$ Without effect	$NR\to10$ $R_0..R_{10}$ Without effect

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. 164

[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, 21, 151

[Barenghi 2009] Alessandro Barenghi, 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

[Barenghi 2010] Alessandro Barenghi, Guido M. Bertoni, Luca Breveglieri, Mauro Pellicioli 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, 151

[Barenghi 2011] Alessandro Barenghi, Luca Breveglieri, Israel Koren and David Naccache. *Fault Injection Attacks on Cryptographic Devices: Theory, Practice and Countermeasures.* In press, 2011. 7, 11, 12, 13, 14, 15

[Bellcore 1996] Bellcore. *New threat model breaks crypto codes.* 1996. 5, 8, 147, 150

[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, 9, 147, 150

[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, 53, 150

[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, 9, 147, 150

[Bruestle 2002] Matthias Bruestle. SOSSE - *Simple Operating System for Smartcard Education.* `http://www.mbsks.franken.de/sosse/index.html`, 2002. 27, 157, 158

[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 FDTC 2005, pages 13–24, 2005. 8, 20, 77, 78, 149, 150, 165, 166

[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, 21, 155

[Denning 1983] Dorothy Elisabeth Denning. Cryptography and data security. Addison-Wesley, 1983. 2, 146

[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, 53

[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, 62

[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, 147

[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, 53, 122, 162

[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, 148

[Joye 2005] M. Joye and F. Olivier. *Side Channel Analysis*. In Encyclopedia of Cryptography and Security, pages 571–576. Kluwer Academic Publishers, 2005. 5, 147

[Kahn 1973] David Kahn. The codebreakers - the story of secret writing. Macmillan, 1973. 3, 147

[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, 147

[Kocher 1998] Paul Kocher. *Introduction to Differential Power Analysis and Related Attacks.* http://www.cryptography.com/public/pdf/DPATechInfo.pdf, 1998. 133

[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, 133

[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, 147

[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, 151

[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, 23

[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, 146

[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, 166

[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`. 31

[NIST 2001] NIST. Announcing the advanced encryption standard (AES) - FIPS 197. National Institute of Standards and Technology (NIST), 2001. 17, 78, 153, 154, 166

[Otto 2004] Martin Otto. *Fault Attacks and Countermeasures*. PhD thesis, 2004. 14, 16, 152

[Paar 2010] Christof Paar and Jan Pelzl. Understanding cryptography. Springer, 2010. 1, 3, 146, 147

[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, 166

[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, 53, 162

[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. 10, 20, 150, 151

[Sanyo 2008] Sanyo. *Sanyo Amorphous Solar Cell - Amorton -* AM-*8801*. `http://semicon.sanyo.com/en/ds_e/ENA0562A.pdf`, 2008. 136, 176

[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. 136, 176

[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, 21, 37, 154

[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, 148

[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, 146, 147

[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, 150

[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 infer secrets. In the context of security characterization of circuits, we have examined practical feasibility of some theoretical models of fault attacks. We used a laser bench as a means of the fault injection.

At the beginning, we performed laser fault injections on a microcontroller implementing an AES cryptographic algorithm. We succeeded to exclude 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, the experiments give a warning for the feasibility of described attacks in the literature by laser. Our tests have demonstrated 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. They also revealed new attack possibilities. Therefore, it conducted us to 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 ces nouvelles attaques de 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. Cela nous a amenés à étudier des contre-mesures adaptées.