Multidimensionality of the models and the data in the side-channel domain

Damien Marion

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

TELECOM ParisTech

2018-ENST-0056

EDITE - ED 130

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

TELECOM ParisTech

Spécialité « Electronique et Communication »

présentée et soutenue publiquement par

Damien Marion

le 5 décembre 2018

Multidimensionality of the Models and the Data in the Side-Channel Domain

Directeur de thèse : **Sylvain Guilley, Professeur invité, TELECOM-ParisTech**

Soutenance prévue devant le jury composé de

M. Louis GOUBIN	Rapporteur, Professeur, Université Versailles-Saint-Quentin-en-Yvelines
M. Jean-Louis LANET,	Rapporteur, Professeur, Université de Limoges, INRIA RBA
M. Guillaume BOUFFARD,	Examinateur, PhD, ANSSI
M. Christophe CLAVIER,	Examinateur, Professeur, Université de Limoges
Mme Annelie HEUSER,	Examinatrice, PhD, CNRS
M. Adrien FACON,	Invité, PhD, Secure-IC

TELECOM ParisTech
école de l'Institut Mines-Télécom - membre de ParisTech
46 rue Barrault 75013 Paris - (+33) 1 45 81 77 77 - www.telecom-paristech.fr

T H È S E

À François Houyvet,
notre bonne vielle Fanfe.

Remerciements

Une thèse dure trois ans, enfin quatre en ce qui me concerne. Quatre ans en termes de rencontres, d'interactions, de découvertes : c'est long. Surtout lorsque l'on réalise une thèse. Peut-être même plus lorsqu'il s'agit d'une thèse CIFRE. Entre le laboratoire (Télécom ParisTech), l'entreprise (Secure-IC), les amis et la famille, cela en fait du monde qui m'a aidé, soutenu, encouragé, supporté... Toutes ces personnes je tiens à les remercier (même si malheureusement, il est certain que j'en oublierai). Car sans eux, jamais je n'aurais réussi à acquérir ce grade de docteur.

Pour commencer, je souhaite remercier l'ensemble des membres prestigieux de mon jury d'avoir accepté de faire partie de celui-ci : Louis Goubin pour la qualité de son rapport. Jean-Louis Lanet, pour son efficacité dans l'urgence ; mais aussi pour ses remarques pertinentes m'ayant permis d'élargir ma vision sur les problématiques liées à la cryptographie en boîte blanche. Je remercie également Guillaume Bouffard pour ses nombreux conseils tout au long de ma thèse via *Hangouts* et pour son amitié fidèle depuis Limoges. Ma reconnaissance va aussi à Christophe Clavier pour m'avoir, lui aussi à Limoges, enseigné et partagé sa passion des canaux auxiliaires. Merci à Annelie Heuser pour avoir été une coautrice de génie, Adrien Facon pour son encadrement de qualité au sein de Secure-IC tout au long de ma thèse. Et pour finir, je remercie Sylvain Guilley en tant que directeur, pour m'avoir fait confiance en me permettant de réaliser cette thèse, pour ses remarques pertinentes, son enseignement et son flot d'idées intarissable.

Ensuite viennent mes collègues et coauteurs de Télécom ParisTech, Olivier Rioul et Nicolas Bruneaux. Je suis heureux d'avoir pu travailler à leurs cotés, depuis Rennes et

ce malgré la distance. Ils m'ont tous deux beaucoup appris et permis de participer à des travaux de grande qualité. À Télécom ParisTech, je me dois aussi de remercier l'équipe administrative : Marianna Baziz, Florence Besnard, Chantal Cadiat et Alain Sibille, pour m'avoir supporté avec tant de patience malgré mon incompétence administrative notoire.

Je tiens aussi à exprimer ma gratitude à Marie Letoret et Hervé Martin de m'avoir sélectionné pour le *French-American Doctoral Exchange Semina (FADEx) 2017* sur la cybersécurité où j'ai pu rencontrer de nombreux doctorants dont les échanges m'ont enrichi et élargi mes connaissances.

Cette thèse ayant eu lieu à Secure-IC (Rennes), il m'est impossible d'oublier mes nombreux collègues qui m'ont subi durant ces quatre années. Sans nul doute que la tâche n'a pas du être facile ! J'ai, à leur coté, beaucoup appris sur la recherche appliquée et les problématiques industrielles. Certains sont devenus de véritables amis avec les années. Je tiens à ce qu'ils sachent que je les remercie toutes et tous. Je souhaite tout de même en citer quelques-uns, ceux ayant répondu présent à l'appel du comptoir (il fallait bien trouver un moyen pour discriminer) : Brice, Cédric, Chloë, Florent, Franz, Janie, Kais, Karine, Margaux, Martin, Michaël, Michel, Olivier, Pierre, Rachid, Romain, Ronan, Sylvie, Théo, Thomas (×2), Thuy, Valentin, Vincent, Yannick, Yannis...

Une thèse étant une expérience pas toujours facile, il faut pouvoir décompresser. C'est pourquoi je remercie toute la team Modjo[1], pour m'avoir accueilli pendant de longues heures de rédactions puis de grimpe. Merci aussi aux grimpeurs : Corentin, Magda, Hugo, Simon et Thomas.

Si une thèse n'est qu'une première étape de la vie d'un chercheur, elle vient achever celle d'étudiant. Tout au long de mes années d'études, j'ai rencontré de nombreux amis exceptionnels. Ils m'ont accompagné bien avant que ne commence ma thèse et ont continué pendant celle-ci. Je ne serais sûrement pas le même sans eux et il sont, chacun à leur manière, pour beaucoup dans cette thèse. Les plus anciens, les *Golbutes* : François, GG (qui nous rapporta Lucie), Savary, Socrate et la Yo. Arrivèrent ensuite les amis de la MPT (Maison Pour Tous) : Elo, la Furte, les Lilous, Mat, les Pagots, Tat, la Togne, Tontons... Puis vinrent les *Almeriasmsus* : Béa, Léa, Loic, Lucia, Lucrezia et Stefano. Pour finir, les amis du master et leurs conjoints, les *Parisiens*, accueillant toujours aussi

[1] www.modjo-escalade.fr

chaleureusement le petit provincial que je suis : Bichon, Floriane, Marie-Luce, Romain, Thomas et Tuc. Merci à vous tous.

Je remercie aussi mon amie, Marion qui pendant mes dernières années de thèse m'a soutenu et empêché d'abandonner et ce malgré la distance.

Pour finir, je tiens à remercier mes parents qui ont toujours été là pour moi, du berceau au labo, avec une telle abnégation que c'est à eux que devrait revenir ce titre de docteur. Je remercie aussi mon frère, pour ses corrections, me permettant de lui dire aujourd'hui : "je suis docteur moi !" ainsi qu'à sa compagne Caro.

Acronyms

Notation	Description	Page List
PC	Program Counter	6, 22, 85
XOR	eXclusive OR	7, 27, 51
AES	Advanced Encryption Standard	7, 8, 11, 12, 14, 26, 64, 78, 93
HW	Hamming Weight	31, 62, 76, 84
SCA	Side-Channel Analysis	8, 9, 26, 28, 45, 66, 87
PCA	Principal Component Analysis	9, 11, 19, 44, 45
LDA	Linear Discriminant Analysis	9, 11, 19, 42, 44, 45
SNR	Signal-to-Noise Ratio	9, 11, 19, 45, 80
DPA	Differential Power Analysis	9, 11, 16, 19–21, 28, 45, 48, 80, 99
CPA	Correlation Power Analysis	10, 34, 68, 88, 111
WBC	White Box Cryptography	10, 35, 86
CTF	Capture The Flag	11, 35, 86
SC	Side-Channel	11, 12, 40, 46, 76, 84, 109
GDB	GNU Debugger	12, 21, 41, 86, 110
NIST	National Institute of Standards and Technology	26
SPN	Substitution-Permutation Network	27

Notation	Description	Page List
IoT	Internet of Thing	28, 109
EM	ElectroMagnetic	28, 83
SCARE	Side-Channel Analysis for Reverse Engineering	29
TA	Template Attack	34, 46, 66
PoI	Points-of-Interest	45
SOSD	Sum-Of-Square Differences	45
ANOVA	ANalysis Of VAriance	46
NICV	Normalized Inter-Class Variance	46
LRA	Linear Regression Analysis	46, 88
SPA	Simple Power Analysis	48
AWGN	Additive White Gaussian Noise	50
HD	Hamming Distance	61, 84
MSE	Mean Square Error	78
DBA	Dynamic Binary Analyzer	84

Convention of Notations

Notation	Description	Page List
X	traces measurements	29, 47, 68, 86, 111
Σ	the noise covariance matrix	31, 47, 68, 109
α	link between the model and the measurements	9, 30, 47, 68, 109
D	number of samples	9, 30, 47, 68, 86, 110
S	model dimensionality	9, 31, 68, 87, 110
$\mathcal{D}(.)$	distinguisher function	11, 34, 47, 68, 96
Pc	Program Counter (PC) in formulas	12, 86
Q	number of traces	12, 29, 47, 69, 87, 111
R	number of registers (in bits)	12, 43, 86
AbsMarg	Absolute distinguishing Margin, proposed by Whitnall *et al.* in [81] and recalled in the *Eq. 5.6*	12, 96
Y	leakage model	15, 29, 47, 68, 87, 110
$k^\star$	the right key	15, 27, 47, 94
N	measurement noise	15, 30, 47, 69
T	plaintext	17, 29, 47, 68, 88
k	key guess	17, 29, 47, 68, 88
argmax (.)	return the argument that produces the highest score of given distributions	17, 35, 47, 68

Notation	Description	Page List
$(.)^\mathsf{T}$	the transpose matrix operator	18, 30, 47, 68
$\mathbb{F}_n$	Galois Field (GF) of order n (for examples $\mathbb{F}_2$ denotes the GF of order 2)	22, 27, 62, 68, 86
$\oplus$	bitwise eXclusive OR (XOR) logic operation	24, 31, 74, 88
S-box	Substitution-box	27, 76, 88
AES-128	Advanced Encryption Standard (AES) with a secret key of 128-bits	27, 61, 88
$\mathbb{R}$	real numbers field	30, 68, 110
$\langle . \mid . \rangle$	scalar product	30
$\|.\|_2$	2-norm (also called Euclidean norm)	30
$\|.\|_F$	Frobenius Norm	30, 71
tr(.)	trace of a matrix	30, 68
$p_X(x)$	Probability function of a random variable X ($p(X)$ in a clear context)	30, 47, 69
$Y^\star$	leakage model corresponding to the right key	31
ϕ	leakage function	31, 51
HW(.)	Hamming Weight (HW) function (sum of weighted bits)	31, 62
$\mathcal{N}(\mu, \Sigma)$	normal distribution of mean μ and covariance Σ	31
$\mathbb{E}(.)$	mean function	33, 57
var(.)	variance function (also denoted σ)	33, 50
R^2	coefficient of determination	46
argmin (.)	return the argument that produces the lowest score of given distributions	48, 68
cov(.)	covariance function (also denoted σ)	56
$\mathbb{Z}$	natural numbers field	86
$\mathbb{N}$	natural positive numbers field	89

Table of Contents

List of Figures

LIST OF FIGURES

List of Tables

CHAPTER 1

Résumé de la Thèse en Français

Contents

1.1 Chapitre 1 : Introduction

Ce manuscrit de thèse débute par une première section décrivant un ensemble de connaissances nécessaires à la bonne compréhension des notions abordées. Une première sous-section décrit en détail l'Advanced Encryption Standard (AES). Cet algorithme cryptographique a fait l'objet d'analyses poussées tout au long de la thèse pour illustrer les contributions proposées. Cette fonction de chiffrement prend en entrée des blocs de 16 octets ainsi qu'une clé secrète. Cette clé secrète est identique pour le chiffrement et le déchiffrement. L'algorithme se décline sous trois formes, chacune d'elle prenant respectivement des clés de 16, 25 et 32 octets. Chacune des ces formes divergent par leurs nombres de tours dont est composé l'algorithme, respectivement 10, 12 et 14. Chacun

des tours de l'AES est divisé en quatre sous-fonctions appliquées successivement sur l'état interne :

- un **OU-exclusif** avec clé de tour, dérivant de la clé secrète ;
- un **décalage par rotation** ;
- une **transformation linéaire** dans un corps de Galois ;
- une **permutation non-linéaire** relative à une table connue publiquement.

Des schémas explicatifs sont fournis dans le manuscrit pour décrire avec précision l'ensemble de ces sous-fonctions.

La seconde sous-section introduit le principe des attaques par canaux auxiliaires, domaine auquel l'ensemble des contributions de la thèse est rattaché. Les attaques par canaux auxiliaires ont été introduites par Kocher *et al.* en 1999 dans l'article intitulé "Differential Power Analysis"[46][1]. Depuis cette date, elles sont connues pour être vecteur d'attaques particulièrement efficaces contre les algorithmes cryptographiques exécutés sur des systèmes embarqués. Ces attaques consistent en l'utilisation d'informations inhérentes à l'exécution de l'algorithme sur un système physique. Ces canaux auxiliaires peuvent être le temps d'exécution, les émanations électromagnétiques, la consommation de courant... Ces attaques ciblant principalement des algorithmes cryptographiques, leur objectif est généralement de retrouver des clés secrètes. Cependant, elles permettent aussi, dans certain cas de réaliser de la rétro-ingénierie de système. Dans le cadre de cette thèse, les contributions proposées sont relatives aux fuites d'informations provenant de la consommation électrique et des émanations électromagnétiques.

Pour faciliter la compréhension, une section introduit l'ensemble des notations utilisées. Ces notations sont les même pour l'ensemble du manuscrits et sont toutes des notations matricielles. Ainsi les fuites d'informations collectées par un attaquant seront stockées dans une matrice noté $X^{D,Q} = \{X_{d,q}\}_{\substack{d<D,\\ q<Q}}$, avec Q le nombre de traces (nombres d'exécutions de l'algorithme attaqué) et D le nombre d'échantillons de ces traces. Les fuites d'informations sont modélisées de la façon suivante :

$$X^{D,Q} = \alpha^{D,S} Y^{S,Q}(k^\star) + N^{D,Q} \tag{1.1}$$

où $Y^{S,Q}(k^\star)$ est la modélisation d'une fuite relative à la manipulation de la donnée secrète $k^\star$. $\alpha^{D,S}$ représente l'influence physique de la manipulation ces données et

[1] analyse différentielle de la consommation de courant, en français

$N^{D,Q}$ le bruit (dont la matrice de covariance est notée $\Sigma^{D,D}$). La *figure 1.1* représente un modèle de fuite théorique avec un bruit nul. Dans ce cas, le modèle est le poids de Hamming, c'est à dire la somme des bits de la valeur manipulée.

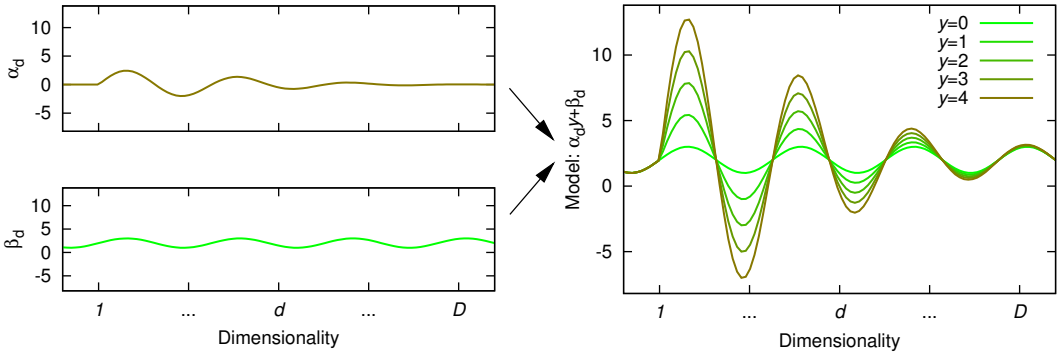


Figure 1.1: Exemple de modèle de fuite avec $S = 2$, Y est le poids de Hamming de valeur de 4-bits et un bruit nul.

La *figure 1.2* illustre un cas pratique où plusieurs modèles se confondent : le modèle poids de Hamming (au début), ainsi que deux modèles plus complexes (au milieu et à droite) où chacun des bits de la valeur manipulée fuit indépendamment.

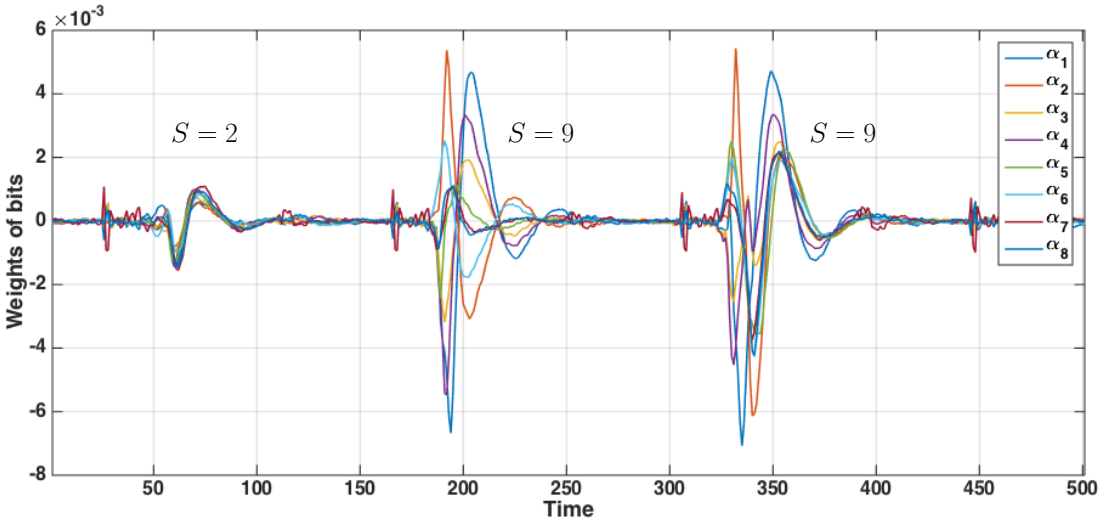


Figure 1.2: Évaluation de fuites de traces réelles fournies lors du DPA contest V4 [74] .

Le modélisation de la fuite est centrale dans l'ensemble des contributions proposées dans le manuscrit. La dimension du modèle S, le nombre d'échantillons de la fuite D,

ainsi que la la nature du bruit N ont fait l'objet d'investigations poussées dans les différents chapitres.

Une fois les fuites X collectées, l'objectif est d'en extraire l'information secrète $k^\star$. Pour se faire, on utilise ce qu'on appelle un distingueur noté $\mathcal{D}$. Il permet de comparer un modèle supposé Y et les données X. Y est fonction de données connues, messages en clair (ou messages chiffrés) noté T et d'une donnée secrète supposée, noté k. Ainsi exprimé, l'objectif du distingueur est de maximiser la probabilité de succès de $D(X, Y = k^\star)$. Cela peut être formalisé de la façon suivante, proposé dans [42] :

$$\mathcal{D}(X^{D,Q}, T^Q) = \underset{k}{\operatorname{argmax}} \left(p(X^{D,Q} | T^Q, k = k^\star) \right). \tag{1.2}$$

L'introduction aux analyses par canaux auxiliaires s'achève par la présentation de différentes attaques. Ces attaques résultent de la participation à une compétition ayant été organisée lors de la conférence CHES-2016[1]. Les analyses proposées ont permis d'atteindre la seconde place du classement étudiant et la vingt-et-unième du général.

L'introduction du manuscrit s'achève par l'explicitation des contributions et la présentation de l'organisation du manuscrit. Celle-ci s'organise en trois chapitres principaux présentant les trois contributions majeures, suivis d'un chapitre de conclusion. La première contribution, intitulée "moins c'est plus" répond à la problématique suivante :

Comment peut-on compresser de façon optimale l'ensemble des points de fuite en un seul échantillon sans perte d'efficacité ?

Ce travail collaboratif avec Nicolas Bruneau, Sylvain Guilley, Annelie Heuser et Olivier Rioul a donné lieu à une publication à CHES-2015 [13].

La seconde contribution intitulée "Fuites Multivariées et Modèles Multiples" répond à la problématique suivante :

Comment peut-on tirer avantage d'un ensemble de points de fuite dans le cadre de modèles multiples ?

Ce travail collaboratif avec les mes auteurs que le précédent a donné lieu à une première publication à PROOF-2016 [14], ainsi qu'à un poster à CHES-2016 puis à la publication d'un version étendue dans le journal *journal of cryptographic engineering* [15].

[1] ctf.newae.com

La dernière contribution intitulée “Analyse Binaire pour L'évaluation des Fuites d'un Code Source” répond à problématique suivante :

Comment peut-on identifier et caractériser des fuites d'informations en exploitant des “traces logiciels” tridimensionnelles ?

Ce travail collaboratif avec Antoine Bouvet, Nicolas Bruneau, Adrien Facon, Sylvain Guilley, Matthieu Lec'Hvien et Thomas Perianin a donné lieu à deux publications, la première à DTIS-2018 [10] et la seconde à SecITC-2018 [32].

L'ensemble des contributions s'organise autour de la multidimensionnalité. La première contribution traite de celle des fuites, dimension D de X. La seconde ajoute à la précédente la problématique de la dimension des modèles, dimension S de Y et α. La dernière, quand à elle, aborde la multidimentionnalité des données en ajoutant une dimension R aux données.

1.2 Chapitre 2 : Moins C'est Plus

Ce chapitre aborde le problème de la réduction de dimension des fuites. En effet il est connu que dans le cadre d'une acquisition de données, les informations sensibles fuitent au cours du temps. L'objectif de la réduction de dimension est de projeter l'ensemble des points de fuite en un seul sans perdre d'information. C'est ce qui est proposé via l'expression “moins c'est plus” : une réduction de dimension et donc une accélération calculatoire sans diminution de la probabilité de succès de l'attaque.

Le résultat principal de ce chapitre est résumé dans le *Théorème 1.1* :

Théorème 1.1. *L'attaque optimale sur des traces multivariées $X^{D,Q}$ est équivalente à l'attaque optimale sur des traces univariées $\tilde{X}^{Q}$, obtenues depuis $X^{D,Q}$ via la formule :*

$$\tilde{X}_q = \frac{(\alpha^D)^\intercal \Sigma^{-1} X_q^D}{(\alpha^D)^\intercal \Sigma^{-1} \alpha^D} \qquad (q = 1, \ldots, Q). \tag{1.3}$$

Dans le cadre de ce chapitre, nous proposons une étude complémentaire de la réduction de dimension en présence de deux modèles de bruit particulier. Le bruit blanc et le bruit autorégressif, dont les formules de projection sont respectivement explicitées

dans les *propositions 1.1* et 1.2.

Proposition 1.1. *Pour le bruit blanc, la réduction de dimension optimale prend la forme suivante :*

$$\tilde{X}_q = \frac{\sum_{d=1}^{D} \frac{\alpha_d}{\sigma_d^2} X_{d,q}}{\sum_{d=1}^{D} \frac{\alpha_d^2}{\sigma_d^2}} \qquad (q = 1, \ldots, Q) \tag{1.4}$$

Proposition 1.2. *Pour le bruit autorégressif, la réduction de dimension optimale prend la forme suivante :*

$$\tilde{X}_q = \tfrac{1}{\sigma^2(1-\rho^2)} \Big[(\alpha_1 - \rho\alpha_2) X_{q,1} + \sum_{d=2}^{D-1} ((1+\rho^2)\alpha_d - \rho(\alpha_{d-1} + \alpha_{d+1})) X_{d,q} + (\alpha_D - \rho\alpha_{D-1}) X_q^D \Big]. \tag{1.5}$$

Un cas réel de bruit localement autorégressif, illustré en *figure 1.3* a été identifié dans des traces de la compétition du DPA contest V2 [73] utilisées pour la validation des résultats.

La méthode de réduction de dimension proposée est ensuite comparée à deux méthodes bien connues de l'état de l'art : l'analyse discriminante linéaire (ALD ou Linear Discriminant Analysis (LDA) en anglais) et l'analyse en composante principale (ACP ou Principal Component Analysis (PCA) en anglais). Les différentes méthodes de réduction de dimension sont comparées à l'aide de la métrique du rapport de signal sur bruit (Signal-to-Noise Ratio (SNR) en anglais). Ces comparaisons ont donné lieu aux *Théorèmes 1.2* et 1.3.

Théorème 1.2. *Le rapport signal sur bruit de l'ACP asymptotique est inférieur à celui de la réduction de dimension optimale*

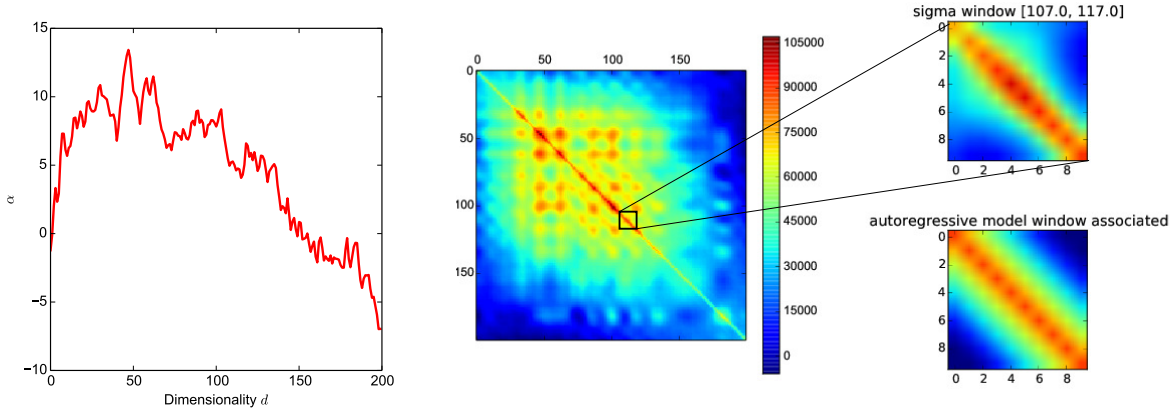


Figure 1.3: Éstimation de $\hat{\alpha}^D$ (*gauche*) et de $\hat{\Sigma}^{D,D}$ (*droite*), avec $Q = 10,000$ traces, illustrant un bruit autorégressif.

Théorème 1.3. *L'ALD asymptotique calcule exactement la réduction de dimension optimale.*

L'ensemble des théorèmes de ce chapitre et du suivant font tous l'objet de démonstrations détaillées.

Pour finir, une validation pratique des résultats est proposée, en premier lieu sur des données simulées, puis sur des traces fournies lors de la compétition du DPA contest V2 [73] .

1.3 Chapitre 3 : Fuites Multivariées et Modèles Multiples

En lien direct avec le chapitre précédent, ce chapitre traite lui aussi de la multidimentionnalitées des fuites. Mais ici, nous allons plus loin et traitons le cas de modèles multiple, lorsque $S \geq 1$. L'ensemble des résultats obtenus dans ce chapitre est résumé en *figure 1.4*. Cette figure explicite les distingueurs optimaux dans le cas de fuites multivariées ($D \geq 1$) et de modèles multiples ($S \geq 1$). Deux cas distincts sont abordés. Le premier, lorsque le modèle (α) est connu. Celui-ci peut-être obtenu lors d'une première étape d'apprentissage. Le distingueur est dans ce cas noté $\mathcal{D}_{\mathrm{ML}}$. Le second, lorsque le

modèle est inconnu, il est alors estimé directement lors de l'attaque, il sera alors noté $\mathcal{D}_{\text{ML,sto}}$.

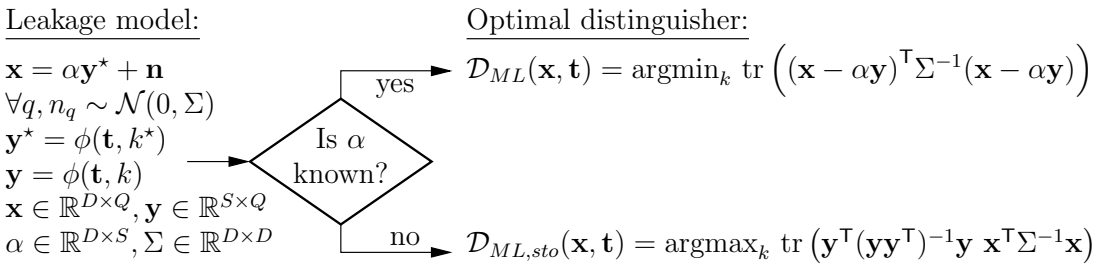


Figure 1.4: Expressions mathématiques des distingueurs optimaux pour des fuites multivariées ($D \geq 1$) et des modèles multiples ($S \geq 1$).

Ce chapitre propose également des algorithmes permettant une implémentation efficace du calcule des différents distingueurs. De plus, une méthode de caractérisation du bruit est également proposée.

Pour finir, plusieurs expérimentations sont proposées pour évaluer l'efficacité des distingueurs proposés. L'une des évaluation est réalisée sur des traces réelles fournies lors de la compétition du DPA contest V4 [74] . Les résultats sont résumés en *figure 1.5*. On y retrouve l'utilisation des distingueurs $\mathcal{D}_{\text{ML,sto}}$ et $\mathcal{D}_{\text{ML}}$. De plus, on y compare l'efficacité du distingueur *Diml* lors de deux cas d'usage. Le premier $\mathcal{D}_{\text{ML}}$ (**self**), lorsque le modèle est parfaitement connu, c'est à dire que α est calculé directement sur les traces analysées. Le second $\mathcal{D}_{\text{ML}}$, lorsque α est calculé sur des trace distinctes.

1.4 Chapitre 4 : Analyse Binaire pour l'Évaluation des Fuites d'un Code Source

Dans ce chapitre, comme dans le précédent, l'analyse porte sur des fuites et des modèles multivariées, mais dans ce cas sur des "traces logiciels" tridimensionnelles. Ces traces sont composées de l'ensemble des données manipulées par un logiciel lors de son exécution. Dans l'objectif d'enregistrer l'ensemble des données manipulées nous utilisons un debugueur (GDB). Ce procédé nous permet d'obtenir l'ensemble les informations suivantes :

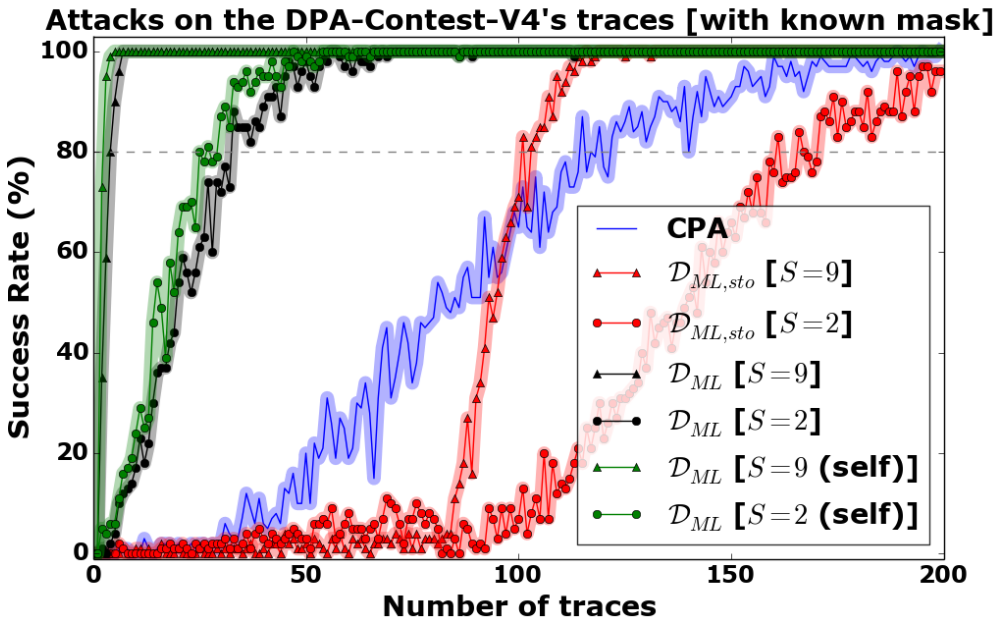


Figure 1.5: Comparaison du taux de succès de l'analyse par corrélation (Correlation Power Analysis (CPA) en anglais), du $\mathcal{D}_{\text{ML,sto}}$, et du $\mathcal{D}_{\text{ML}}$ avec $S \in \{9, 2\}$

- les seize registres de 64-bits : `rax`, `rbx`, `rcx`, `rdx`, `rsi`, `rdi`, `rbp`, `rsp`, `r8`, `r9`, `r10`, `r11`, `r12`, `r13`, `r14`, `r15`,
- les six registres de 16-bits : `cs`, `ss`, `ds`, `es`, `fs`, `gs`,
- les 64-bits du registre de drapeaux,
- le pointeur d'instruction, noté PC (Program Counter en anglais).

Les informations listées correspondent à une exécution sur une architecture x86 de 64-bits. Les données ainsi récoltées sont stockées, comme dans les chapitres précédents, dans une matrice noté $X^{D,R,Q} \in (\mathbb{F}_2)^{D,R,Q}$, avec D le nombre de changements du pointeur d'instruction, R le nombre de bits de registres enregistrés et Q le nombre de traces. Un exemple de *trace logiciel* est illustré en *figure 1.6*. Cette trace provient de l'exécution d'un algorithme cryptographique en boîte blanche fourni lors de la compétition de CHES-2016. En blanc, sont représentés les bits à zéro et en noir les bits à un.

Une fois les données enregistrées, nous proposons dans ce chapitre deux algorithmes de prétraitement permettant une analyse efficace.

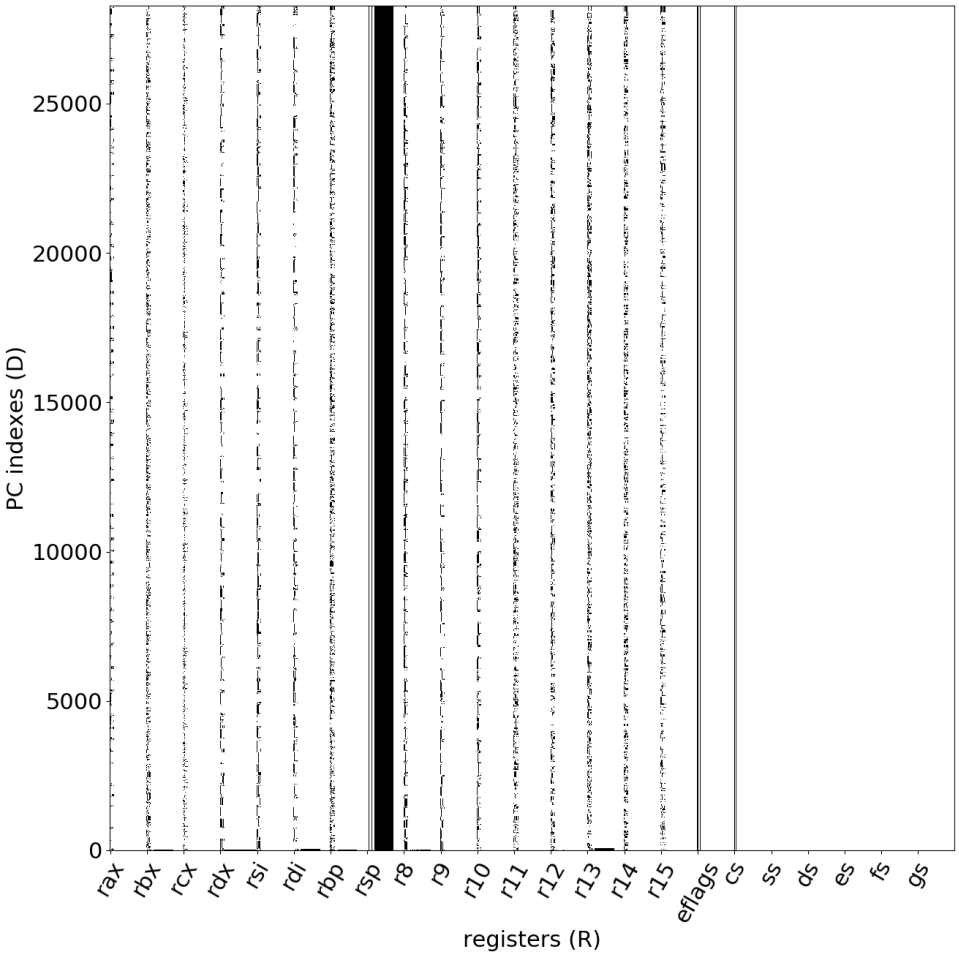


Figure 1.6: Exemple de *trace logiciel* de l'exécution d'un algorithme cryptographique en boîte blanche

Un premier algorithme permet de resynchroniser les traces. En effet, la présence de choix conditionnels dans les implémentations peut engendrer des désynchronisations entre les différentes traces. Si des traces ne sont pas alignées il devient alors impossible des les analyser. L'algorithme proposé permet de resynchroniser rapidement les données en se basant uniquement sur les différentes valeurs du pointeur d'instructions.

Un second algorithme permet, quand à lui, de sélectionner les points pouvant révéler

de l'information. En effet, de nombreux éléments enregistrés ne sont pas pertinents pour l'analyse. L'algorithme, dit de débruitage, utilise deux matrices particulières. La première, la matrice d'activité 1.1, permet de déterminer l'ensemble des points invariants relativement à l'axe Q.

Definition 1.1. ***La matrice d'activité*** $A^{D,R}$ *d'un ensemble de données* $X^{D,R,Q}$ *est définie comme :*

$$A^{D,R} = \left[A_{d,r} = \begin{cases} 1, & if \sum_{q=0}^{Q-1} X_{d,r,q} \notin \{Q, 0\} \\ 0, & sinon \end{cases} \right]_{\substack{d<D, \\ r<R}}$$

La seconde est la matrice de transition 1.2, elle permet de déterminer les invariants d'un ensemble de données relativement à l'axe D.

Definition 1.2. ***La matrice de transition*** $T^{D,R}$ *d'un ensemble de données* $X^{D,R,Q}$ *est définie comme :*

$$T^{D,R} = \left[T_{d,r} = \begin{cases} 1, & if\ d = 0 \\ \vee_{q=1}^{Q} X_{d-1,r} \oplus X_{d,r} & sinon \end{cases} \right]_{\substack{d<D, \\ r<R}}$$

L'utilisation de l'algorithme de débruitage permet de fortement diminuer l'ensemble des donnés à analyser. En effet, l'application de cet algorithme à l'implémentation cryptographique en boîte blanche de CHES-2016 permet d'obtenir les résultats illustrés en *figure 1.7*. On observe ainsi que le débruitage permet d'identifier les 0.29% des données de départ pouvant fuiter de l'information.

Une fois les données resynchronisées et débruitées, une analyse par corrélation permet de retrouver les clés secrètes d'implémentations cryptographiques en boîte blanche. De plus, l'analyse proposée permet, grâce aux pointeurs d'instructions, de retrouver les lignes de codes et les registres d'où proviennent les fuites.

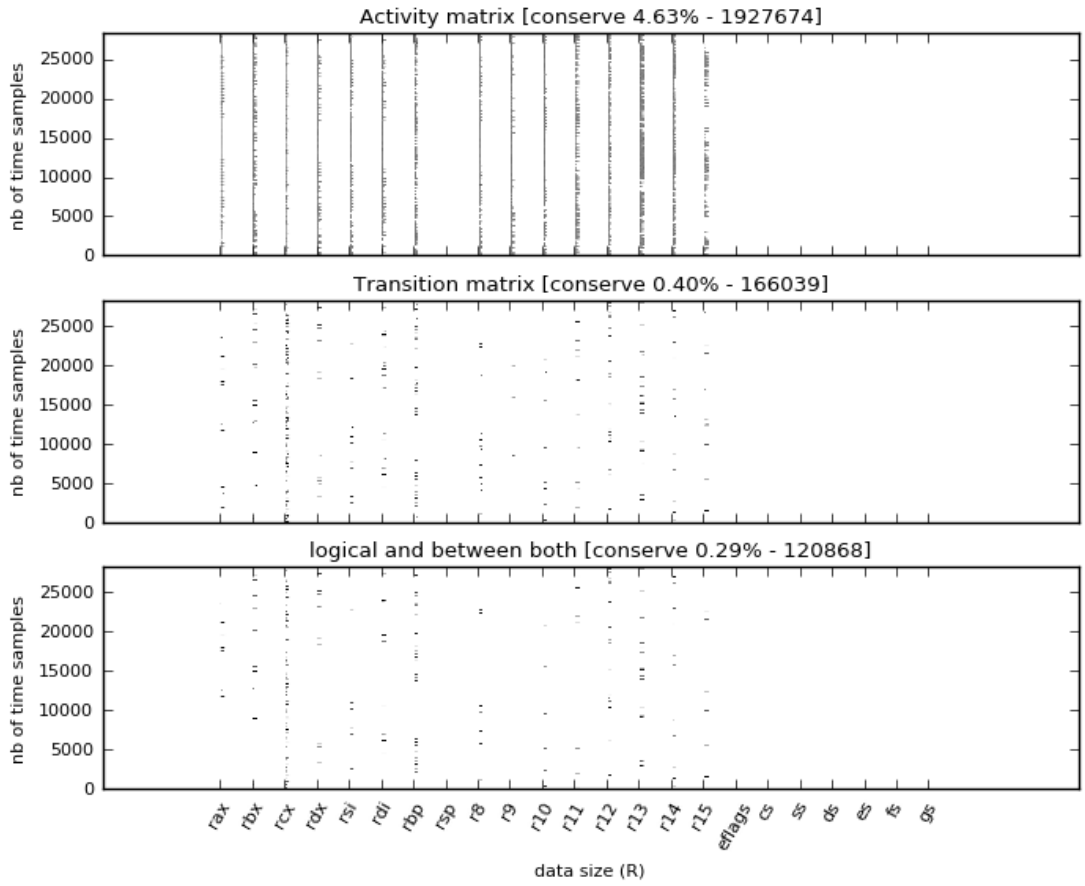


Figure 1.7: Illustration des matrices $A^{D,R}$ (en haut),$T^{D,R}$ (au milieu) et $A^{D,R} \wedge T^{D,R}$ (en bas). Les 0 sont représentés en blanc, les 1 en noir.

CHAPTER 2

Introduction

Contents

2.1 Prior Knowledge

2.1.1 Advanced Encryption Standard (AES)

The Advanced Encryption Standard (AES) is a symmetric cryptographic algorithm, initially called Rijndael, submitted to the National Institute of Standards and Technology (NIST) competition in 1998 and ratified as a standard in 2001 [1]. The algorithm is a block cipher that digests input blocks of 128 bits. The internal state is also made up of 128 bits, where the numbering is visible in *Fig. 2.1*. The input master key is sized

128, 192 or 256 bits, according to the wanted level of security, and is used to generate the 11, 13 or 15 rounds keys. Indeed, the AES falls in the Substitution-Permutation Network (SPN) algorithms with 10, 12 or 14 rounds as a function of the key size. Each round is divided in four subfunctions:

- **the add round key** applies a eXclusive OR (XOR) between the current state and the subkey round. The round keys are computed applying the Rijndael keyschedule to the input key (also called master key),
- **the S-box** is publicly known 8-bit substitution function with good non-linearity properties,
- the **shiftRows** lets the first row unmodified, shifts the second one by one to the left, the third one by two and the last one by three as displayed in the following *Fig. 2.1*,

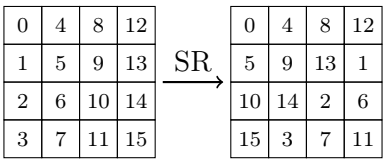


Figure 2.1: ShiftRows' effect on the internal state of an AES[2].

- **the mixColumns** is a linear function applied on each column of the input state, is represented as matrix product in the *Fig 2.2*. For examples, $o_0 = \texttt{0x02}.i_0 + \texttt{0x03}.i_1 + i_2 + i_3$, the operation are done in $\mathbb{F}_{2^8}$.

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \cdot \begin{bmatrix} i_0 & i_4 & i_8 & i_{12} \\ i_1 & i_5 & i_9 & i_{13} \\ i_2 & i_6 & i_{10} & i_{14} \\ i_3 & i_7 & i_{11} & i_{15} \end{bmatrix} = \begin{bmatrix} o_0 & o_4 & o_8 & o_{12} \\ o_1 & o_5 & o_9 & o_{13} \\ o_2 & o_6 & o_{10} & o_{14} \\ o_3 & o_7 & o_{11} & o_{15} \end{bmatrix}$$

Figure 2.2: MixColumns' effect on the internal state of an AES[2]. The product in $\mathbb{F}_{2^8}$ is symbolized here by a bold dot '$\cdot$'.

The complete subfunctions' schedule of an AES-128 is provided in *Fig. 2.3*, it starts off by a XOR between the 128-bit of the plaintext and the master key ($k^\star{}_0$) ; followed by ninth rounds composed of the four subfunctions, previously described, and a last one without the *mixColumns*. Almost all practical validations we present are done on

[2]figure inspired by [43]

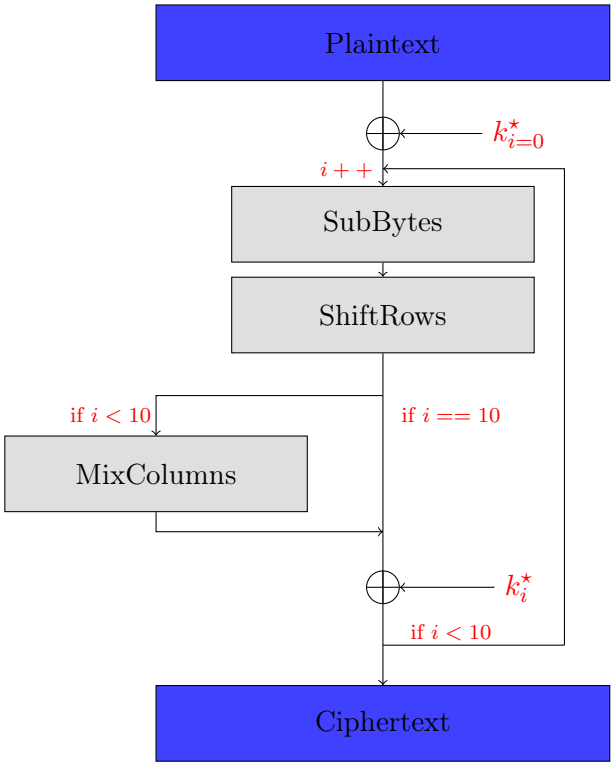


Figure 2.3: AES complete schedule.

the AES implementation. This choice is motivated by its robustness, as it remains cryptographically secure after more than twenty years of evaluation ; and by its wide use in many sorts of applications. Further more, the implementations and the set of traces used are publicly available allowing the repoductibility.

2.1.2 Introduction to the Side-Channel Analysis (SCA)

Since ′99 and the seminal paper of Kocher *et al.* [46] called Differential Power Analysis (DPA), embedded systems (smartcards, smartphones, Internet of Thing (IoT)...) are known to be vulnerable to the Side-Channel Analysis (SCA). This kind of attacks use leakage of information from non-conventional communications channels (ElectroMagnetic (EM), execution time, power consumption...) to extract secret information. The main application of the SCA is the recovery of secret keys manipulated by symmetric or asymmetric cryptographic algorithms.

Others exploitation of the SCA have been proposed as the Side-Channel Analysis for Reverse Engineering (SCARE) [19, 21, 55, 64], with the objective to reveal secrets characteristics of the implementation. For examples SCARE could be used to recover secret S-box. Indeed, it could happen that manufacturers secretly change functions of public cryptosystem even if Kerckhoffs's principle advises that a cryptosystem "must not require secrecy" (except the key) [45] to be secure.

In the case of key recovering, the objective is to discriminate $k^\star$ between k comparing recorded leakage X and models Y. But k have been presented as an n-bit vector (usually, $n = 8$) that is much more less than a real cryptographic key size. Indeed, some institutions, summarized in *Tab. 2.1*, advise to use a key length between 84-bit and 3072-bit according to the chosen cryptographic primitives. That is why the SCA adopts "divide-and-conquer" approach, splitting the entire key in words of n-bit k, with n enough small to be exhaustively iterated. Then each sub key is independently analyzed.

Method	Date	Sym.	Factoring Modulus	Discrete Logarithm		Elliptic Curve	Hash
				Key	Group		
Lenstra/Verheul[48]	2018	84	1771 (1376)	149	1771	158	168
Lenstra Updated[47]	2018	80	1329 (1478)	160	1329	160	160
ECRYPT II[31]	2016/20	96	1776	192	1776	192	192
NIST[6]	2016/30	112	2048	224	2048	224	224
ANSSI[4]	2014/20	100	2048	200	2048	200	200
IAD-NSA[56]	-	256	3072	-	-	384	384
BSI [16]	2017/22	128	2000	250	2000	250	256

Table 2.1: Key size recommendations from national agencies, academic and industrial groups[3].

For example, in the case of the AES-128 , the 128-bit of the secret key are generally split in sixteen 8-byte words. In this way, to recover the whole secret, X is compared to $\{Y_b^Q(k)\}_{\substack{b<16 \\ k<256}}$. But in order to simplify the notations and without loss of generality, the formula are usually given only for one n-bit words, so $k^\star, k$ and T denote n-bit words and the generalization to the entire key recovering is trivial.

[3]Data gathered from www.keylength.com and all key sizes are provided in bits.

2.2 Notations

2.2.1 Matrices

Whatever the target and regardless of the acquisition method, an attacker will record traces and additional data as inputs and/or outputs values. Matrix is the perfect mathematical object to store and manipulate these collected data. That is why we adopt in the whole thesis matrix notations. The queries are indexed by $q = 1, \ldots, Q$, where Q is the number of traces. The samples in a given trace are indexed by $d = 1, \ldots, D$. Any matrix containing D samples from Q queries is denoted by:

$$M^{D,Q} = (M_{d,q})_{\substack{d<D,\\ q<Q}},$$

where $d = 1, \ldots, D$ is a row index and $q = 1, \ldots, Q$ is a column index. We also denote all dth samples for all traces as $(M_{d,q})_{q<Q} = {M^Q}_d$, and all the samples for the qth trace as $(M_{d,q})_{d<D} = M_q^D$. Thus, M_d^Q is a row vector and M_q^D is a column vector. Two matrices noted side-by-side are implicitly multiplied.

The notation $(\cdot)^\mathsf{T}$ is for transpose. For instance, if $u = u^D$ is $D \times 1$ matrix, then $u^\mathsf{T} = (u^D)^\mathsf{T}$ is a $1 \times D$ matrix. The usual scalar product on $\mathbb{R}^D$ is denoted by $\langle u|v\rangle = u^\mathsf{T} v \in \mathbb{R}$. The 2-norm (also called Euclidean norm) of u is $\|u\|_2 = \sqrt{\langle u|u\rangle}$. Then let $\|.\|_F$ denote the Frobenius norm of a matrix (square root of the sum of its squared elements), such that $\|M\|_F = \sqrt{\mathrm{tr}\,(MM^\mathsf{T})}$ where we denote by $\mathrm{tr}\,(\cdot)$ the trace of a square matrix, that is the sum of its diagonal terms. Note that $\mathrm{tr}\,(AB) = \mathrm{tr}\,(BA)$ for compatible matrix dimensions.

Random variables will be denoted by capital letters. The probability density function of a random variable X, as a function of x, is denoted by $p_X(x)$ or simply $p(x)$ if the context is clear.

2.2.2 Signals

Let X denote the leakage measurements, Y the model, N the measurement noise, and α the link between the model and the measurements. Notations X, Y are consistent

with the usual convention in machine learning, where X is for the collected data and Y for the classification labels. The model Y depends on a key guess k, an n-bit vector (as usual $n = 8$), and on some known plaintexts T (usually also an n-bit vector, it could also be the ciphertext). In a view not to overload the notations, we write Y instead of $Y(k)$. As it is customary in SCA, the correct key is denoted by $k^\star$. The corresponding model using the correct key $Y(k^\star)$ is denoted by $Y^\star$. A sensitive variable that depends on the unknown secret key k is leaking through a leakage function ϕ. Let S be the model dimensionality and $\phi : {\mathbb{F}_2}^n \times {\mathbb{F}_2}^n \to \mathbb{R}^S$ a vectorial function, with S components. Typically, ϕ is the Hamming Weight (HW) function, a sum of weighted bits, or its composition with a S-box. In order to further simplify the mathematical derivations, we assume that ϕ is centered. The model for a given key byte hypothesis k is given by

$$Y_q(k) = \phi\ (T_q \oplus k), \tag{2.1}$$

a well-known example is $Y = \mathrm{HW}\ (T \oplus k)$, where HW is the Hamming Weight function. The actual leakage can be written as

$$X_{d,q} = \alpha_{d,S} Y_{S,q}(k^\star) + N_{d,q}, \tag{2.2}$$

where the weights $\alpha_{d,S}$ are not all zero, and $N_{d,q}$ is some random measurement noise. In matrix notation, we can summarize the equations for different values of d and q by a single matrix equation

$$X^{D,Q} = \alpha^{D,S} Y^{S,Q}(k^\star) + N^{D,Q} \tag{2.3}$$

where $\alpha^{D,S}$ is a S-column matrix and $Y^{S,Q}(k^\star)$ is a S-row matrix, whose product is a $D \times Q$ matrix. Notice that our convention to consider traces as lines and dimensions as rows allows us to write the deterministic part of the leakage as $\alpha Y^\star$ which writes more naturally than the opposite order where traces would be viewed as a vertical time series. We make the stationary assumption that the noise distribution does not depend on the particular query, that is, the N_q^D are independent and identically distributed independently of the value of q. For a given q, however, the noise samples of N_q^D can very well be correlated. We assume that N_q^D follows a D-dimensional zero-mean multivariate normal distribution $\mathcal{N}(0, \Sigma,)$, where covariance matrix Σ is a symmetric positive definite $D \times D$ matrix. Therefore, there exists a matrix $\Sigma^{1/2}$, which is such that $\Sigma^{1/2}\Sigma^{1/2} = \Sigma$. We refer to $\Sigma^{1/2}$ as the standard deviation noise matrix.

2.2.3 Models Illustrations

With the aim of illustrating the signals notations previously introduced, we provide some concrete examples for distinct values of S. The leakage signal may be represented as a continuous curve as illustrated in *Fig. 2.4*. The practical acquisition is done through a temporal series of D "discrete samples" within one clock period. For $S = 1$, the traces

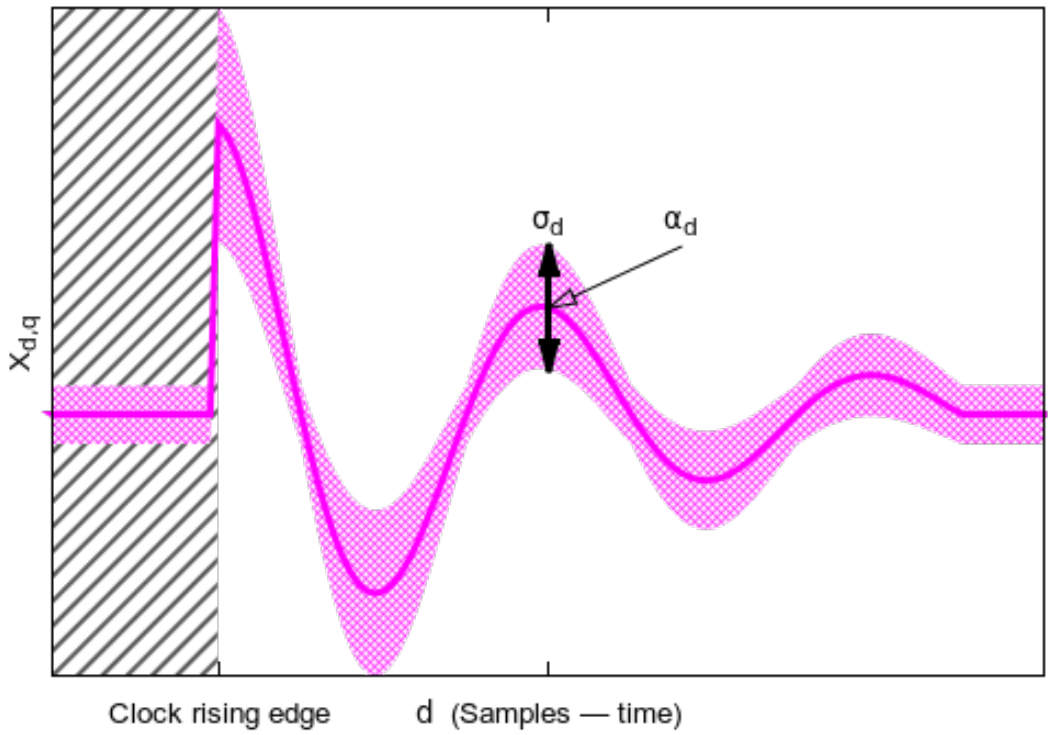


Figure 2.4: Example of a modulated trace X_q^D

consist only in a modulation of the model plus noise as in [12, 13]. When considering traces that are not only modulated but also have an offset term we have $S = 2$. We then write the 2-dimensional model as $\binom{Y}{1^Q}$, where Y and 1^Q are $1 \times Q$ matrices $(Y_1, Y_2, \ldots, Y_Q)$ and $(1, 1, \ldots, 1)$. The $D \times 2$ matrix α in *Eq. 2.3* actually takes the special form $(\alpha\ \beta)$ where β is the offset value. An illustration is provided in *Fig. 2.5* where the parameter $\beta \in \mathbb{R}^D$ is the waveform when there is no signal, whereas $\alpha \in \mathbb{R}^D$ is the signal envelope. The complete model is the sum $\alpha Y + \beta$, where Y is the HW of some intermediate variable (such as the XOR operation $T \oplus k^\star$) on $n = 4$ bits. While the leakage signal may be represented as a continuous curve as illustrated in *Fig. 2.5*,

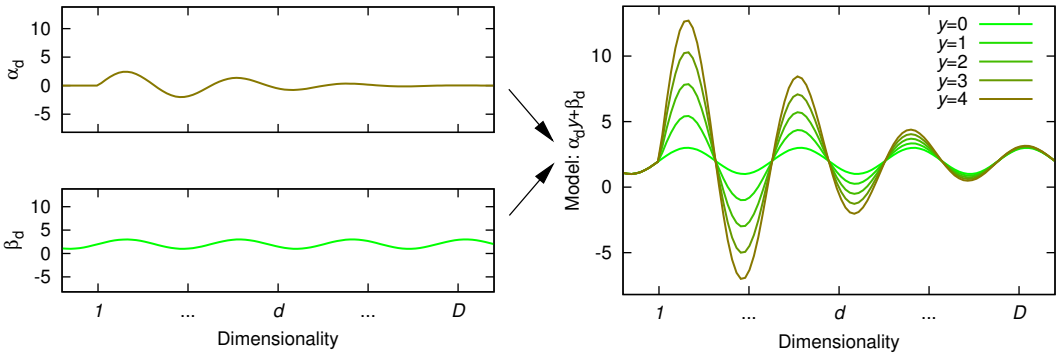


Figure 2.5: Example of leakage model with $S = 2$, Y is the HW of 4-bits values and no noise is added

the practical acquisition is done through the temporal series of D "discrete samples", typically within one clock period. For $S = 2$, we thus write *Eq. 2.3* as

$$X = \alpha Y^{\star} + \beta 1^{Q} + N \tag{2.4}$$

where X is $D \times Q$, α and β are $D \times 1$, $Y^{\star}$ and $1^{Q} = (1, \ldots, 1)$ are $1 \times Q$, and N is $D \times Q$. Here Y is assumed centered: $\mathbb{E}(Y) = 0^{Q} = (0, \ldots, 0)$ (since the non-centered part is captured by the $\beta 1^{Q}$ term) and of unit variance for every q: $\mathrm{var}(Y_q) = \mathbb{E}(Y_q^2) = 1$. For $S \geq 2$, the actual value of S reflects the complexity of the model. For example, in the *weighted sum of bits* model, the model for each trace can be written as $\sum_{s=1}^{n} \alpha_s Y_s + \beta$, where Y_s is the s^{th} bit of the n-bit sensitive variable Y. Accordingly, we have

$$\begin{aligned} & S = n + 1, \qquad \text{and thus:} \\ \alpha = \begin{pmatrix} \alpha_1 & \ldots & \alpha_n & \beta \end{pmatrix}, & \qquad Y = Y_1 \ldots Y_n \, 1^{\mathsf{T}}. \end{aligned} \tag{2.5}$$

This leakage model is more complex than before but may arise in practice. For example, *Fig. 2.6* plots the coefficients $\alpha_1, \ldots, \alpha_8$ estimated of the traces taken from an ATMega smartcard—the datasets are available from the DPA contest V4 [74] team. In particular one can observe that samples around $[50, 80]$ are ordered by HW: this part of the trace resembles the upper left part of *Fig. 2.5* for $S = 2$. By analyzing the $(n+1)$-variate model of *Eq 2.5*, one can indeed see that around $[50, 80]$, the vectors $\alpha_1, \ldots, \alpha_8$ are almost identical. However, samples in intervals $[170, 250]$ or $[330, 400]$ have a more complex model. These times, the eight vectors $\alpha_1, \ldots, \alpha_8$ are clearly different, so the leakage is 9-variate.

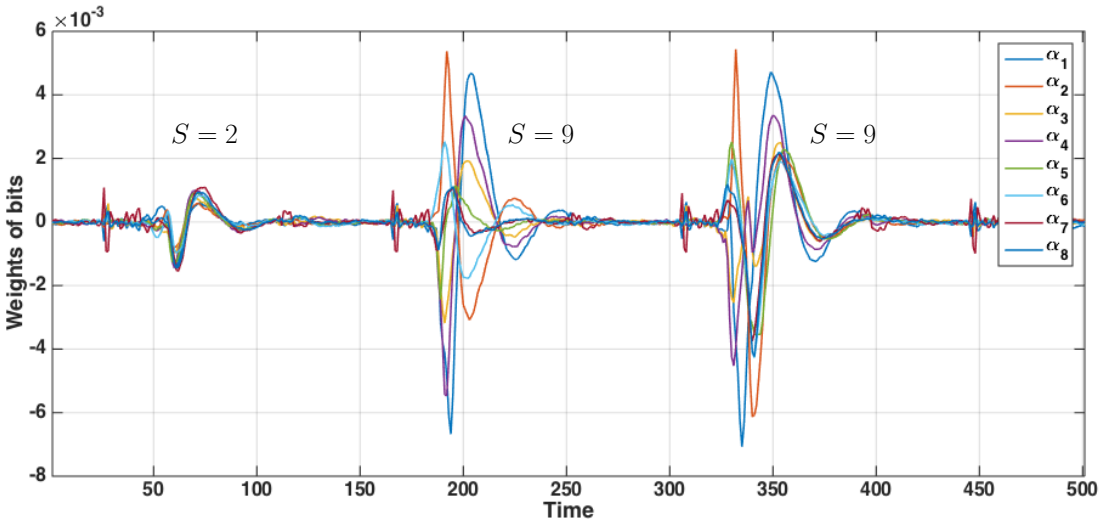


Figure 2.6: Leakage evaluation of traces from the DPA contest V4 [74] (knowing the mask)

2.3 Distinguishers

Once we have the collected the data X, and the models Y, the comparing step use a discriminant function called a *distinguisher* and noted $\mathcal{D}$. For example, the DPA [46] uses the difference between averaged traces, the Correlation Power Analysis (CPA) [11] the Pearson correlation coefficient, the Template Attack (TA) [18] the probability density function of Gaussian distributions... A distinguisher $\mathcal{D}$ maps a collection of leakages X and publicly known plaintexts (or ciphertexts) bytes T to an estimation of the secret key $k^\star$. Thereafter, Heuser *et al.* define in [42] the notion of *optimal distinguisher* rewriting the SCA as a communication channel problem. The goal is to maximize the probability of success of $\mathcal{D}(X, Y = k^\star)$ and it could be formalized as in the *Theorem 2.1* proposed in [42].

Theorem 2.1 (Optimal distinguishing rule)**.** *The optimal distinguishing rule is given by the* maximum a posteriori probability (MAP) *rule*

$$\mathcal{D}(X^{D,Q}, T^Q) = \underset{k}{\operatorname{argmax}} \Big(\mathbb{P}\{k\} \cdot p(X^{D,Q} | T^Q, k = k^\star) \Big). \tag{2.6}$$

If the keys are assumed equiprobable, i.e., $\mathbb{P}\{k\} = 2^{-n}$*, Eq.* (2.7) *reduces to the* maximum likelihood (ML) rule

$$\mathcal{D}(X^{D,Q}, T^Q) = \underset{k}{\operatorname{argmax}} \Big(p(X^{D,Q} | T^Q, k = k^\star) \Big). \tag{2.7}$$

In the whole thesis we consider the attacker does not inject partial information gathered from the leakage analysis into a possible choice of T (nonadaptive attack). The presented results tolerate chosen texts attacks, but consider them only as observed inputs. We do not optimize the attack according to chosen inputs. Thus $Y_1, Y_2, \ldots, Y_Q$ are assumed i.i.d. (denoted by Y). Under the adopted leakage model it follows that the leakage measurements $X_1, X_2, \ldots, X_Q$ are also i.i.d. (denoted by X).

2.4 Examples of Side-Channel Analysis (SCA)

To conclude the introduction of the SCA and to emphasize the efficiency of this kind of analysis we provide in the current section a short presentation of the Capture The Flag (CTF) of CHES-2016[4]. This Challenge have been subject of our participation during the second year of the thesis for which we finish at the second rank of the student participants and at the 24th (over 79) for the overall ranking. During the CTF two types of cryptographic implementations was submitted:

1. AES-128 running on Atmel XMEGA, with power traces and the secret key not stored in firmware (attack via DPA).
2. AES-128 running on Linux computer, without power traces and the secret key stored in firmware (White Box Cryptography (WBC) implementation).

[4] ctf.newae.com/

We focus here on the analysis of the provided power traces, an analysis of the WBC challenge is provided in the *Chap. 5*. For each power traces challenge, six distinct datasets are provided, four with a known key(s) to let the attacker realizing learning steps and two with the same secret key to recover. All datasets are composed of $Q = 1000$ traces with the corresponding plaintexts and ciphertexts. The content of the datasets are given below:

- **for the known key datasets:**
  - "knownfixed-fixed": the key and the plaintext are fix,
  - "knownfixed-rand": the key is fixed and the plaintexts are random,
  - "knownrand-fixed": the keys are random and the plaintext is fix,
  - "knownrand-rand": the keys and the plaintexts are random,
- **for the unknown key datasets:**
  - "secretfixed-fixed": the key and the plaintext are fix,
  - "secretfixed-rand": the key is fix and the plaintexts are random.

The attacks that we proposed in this section only need the "secretfixed-fixed" datasets, and the plaintexts to be mounted. We provide in the following *Subsec. 2.4.1* a complete explanation of the analysis mounted to recover the secret key of the first submitted implementation. Then we summarize in the *Subsec. 2.4.2* the attacks' methodologies that we used to recover the secret key of some other challenges.

2.4.1 First Key Recovering

The first submitted implementation is described as "A very straight-forward AES-128 implementation written in C[5], Standard CPA attack should work". In fact the CPA focusing the output of the S-box at the first round, on which we apply the HW function, provides very efficient results. Formalizing, we get the following *Formula 2.8* for the leakage model (we do not precise the model dimensionality because is equal to one, $S = 1$):

$$Y^{Q,K,B} = \{Y_{q,k,b} = \text{HW}\ (\text{S-box}\ (T_{q,b} \oplus k))\}_{\substack{k<256 \\ q<Q, b<16}} \tag{2.8}$$

For the CPA, the distinguishing step is the Pearson correlation coefficient that we compute between $X^{D,Q}$ and $Y^{Q,K,B}$. In the *Fig. 2.7* we summarized the obtained re-

[5]For all the submitted implementations the source code is provided

sults. In grey we display the best correlation scores obtained with the bad guesses: $\left\{\max_{k_b<256, k_b\neq k^\star_b}(\mathcal{D}(X^{D,Q}, Y^Q_{k_b,b}))\right\}_{b<16}$ and in red the result obtained with the right key: $\left\{\mathcal{D}(X^{D,Q}, Y^Q_{k^\star_b,b})\right\}_{b<16}$. The presented results have been obtained with only one hundred traces ($Q = 100$) and focusing the first two thousand samples ($D = 2000$). The x-axis represents the time samples (D) and the y-axis the absolute value of the Pearson correlation coefficients.

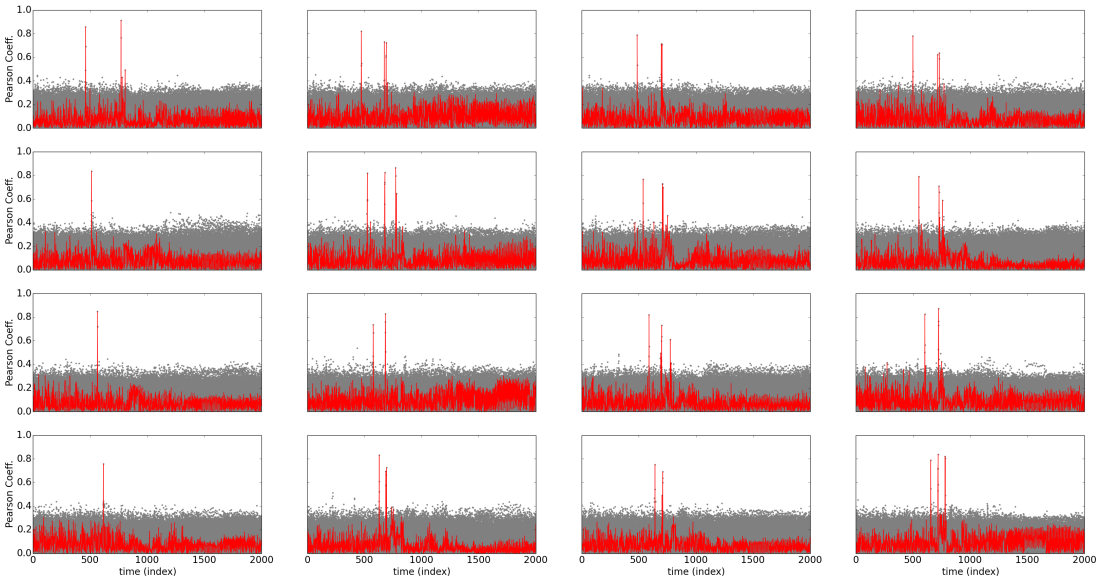


Figure 2.7: Results of the CPA for the first CTF Challenge.

The entire dataset is not needed to recover the entire secret key. We can see in the *Fig. 2.8* the rank of the right key byte, the x-axis represents the number of traces used for the CPA and the y-axis gives the corresponding rank of the right key byte. We show that all the key bytes are recovered in less than forty traces.

The proposed CPA permits us to recover the following secret key:

```
0x7b 0x56 0x27 0xfa 0x8e 0x4 0x8b 0x57 0x90 0xcd 0xe1 0xdd
0xd9 0x18 0x1d 0x1f
```

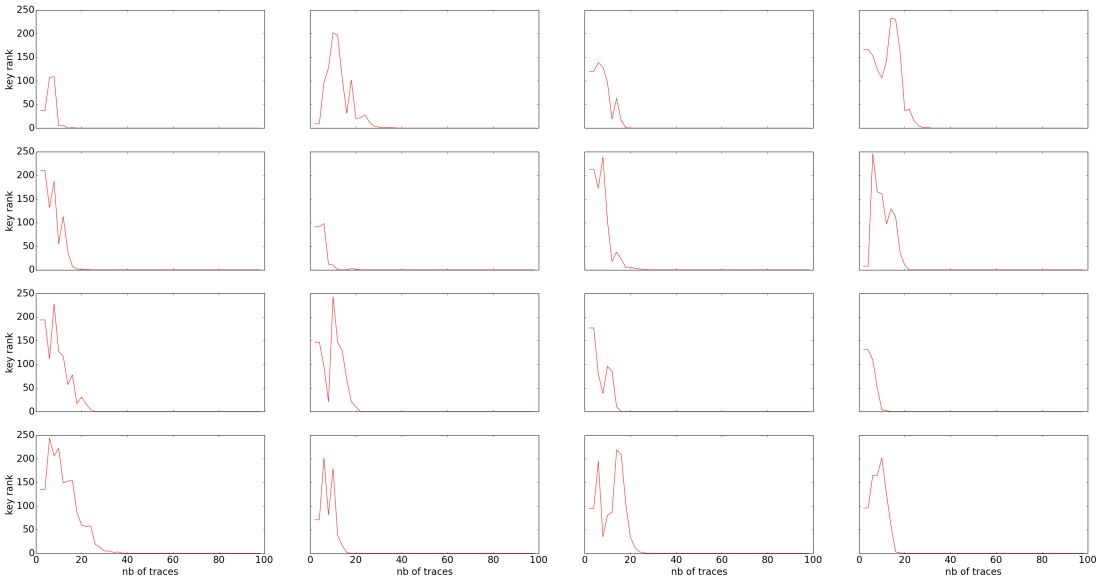


Figure 2.8: Ranks (over 256) of the right key in function of the number of traces used by the CPA for the first CTF challenge.

2.4.2 Successfully Mounted Attacks

The following *Listing 2.4.2* provides the description of six other attacks that lead us to recover the entire secret key. For each exploit we provide the name of the challenge in bold, a short description of the countermeasure(s) used in the implementation under attack and the methodology of our attack.

- **Stagegate #2:** "AES in C with a tiny bit of random jitters before the encryption happens":
  1. We resynchronize the dataset on a recurrent pattern.
  2. We realize a CPA on the resynchronized traces with the models given in the *Formula 2.8*.
- **AES RSI:** "AES with Random Starting Index shuffling countermeasure on S-box":
  1. We identify the sixteen S-box computation at the first round: $X^{D,Q} -> \{X_b^{D'=30,Q}\}_{b<16}$, where $X_{b,q}^{D'=30}$ is the power leakage of one of the sixteen S-box computation in the trace q. The shuffling countermeasures on S-box prevents to identify, for a trace q_0 and a byte $b_0 < 16$, the corresponding

power leakage of S-box($T_{b_0,q_0} \oplus k^\star{}_{b_0}$) in $\{X^{D'=30}_{b,q_0}\}_{b<16}$.

2. We compute the sum of the sixteen S-box computations: $\{X^{D'=30,Q}_b\}_{b<16} -> X^{D'=30,Q}_{sum} = \sum_{b<16} X^{D'=30,Q}_b$.
3. We realize a CPA on the summed traces and the models given in the *Formula 2.8*.

- **Confusion:** "AES with lots of jitters (dummy operation in all the implementation)":
  1. We identify the sixteen S-box computation in each trace. To achieve this we compute the cross-correlation between the power leakage of a manually extracted S-box computation pattern and all the traces. Indeed, we get $X^{D,Q} -> \{X^{D',Q}_b\}_{b<16}$.
  2. We realize a CPA on the sixteen S-box executions and the models given in the *Formula 2.8*: $\{\mathcal{D}(X^{D',Q}_b, Y^Q_{k,b})\}_{b<16,k<256}$.
- **AES with lots of jitters:** "Who needs masking... (shuffling of the S-box computation)":
  1. We identify each S-box computation using a recurrent pattern: the huge peak that appears at each S-box computation.
  2. We compute the sum and the CPA as in the **AES RSI**.
- **FdLSifu1:** "Shuffle and partial random (random S-box execution)": exactly the same attack as for **AES RSI**.
- **Plebe1:**"Almost state-of-the-art AES (masked and shuffled S-box)":
  1. We identify the masking scheme described in *Figure 2.9*.
  2. We visually identify the power leakage of the Leak 1: $\{X^{D',Q}_{\text{state},i}\}_{i<4}$ and of the Leak 2: $\{X^{D',Q}_{\text{mask},i}\}_{i<4}$, as we can see in the *Fig. 2.10*.
  3. We apply the high order CPA proposed in [60], with the following datasets (centered products, where $\overline{X^{D'}_{\star,i}} = (\frac{1}{Q}) \sum_{q<Q} (X^{D'}_{\star,i,q})$):

$$\{X^{D',Q}_i\}_{i<4} = \{(X^{D',Q}_{\text{mask},i} - \overline{X^{D'}_{\text{mask},i}}) \times (X^{D',Q}_{\text{state},i} - \overline{X^{D'}_{\text{state},i}})\}_{i<4},$$

and the following model:

$$\{Y^Q_{k,b} = \text{HW}(3.\texttt{tmp}_b) + \text{HW}(2.\texttt{tmp}_{b,k}) + \text{HW}(\texttt{tmp}_{b,k}) + \text{HW}(\texttt{tmp}_{b,k})\}_{b<16,k<256},$$

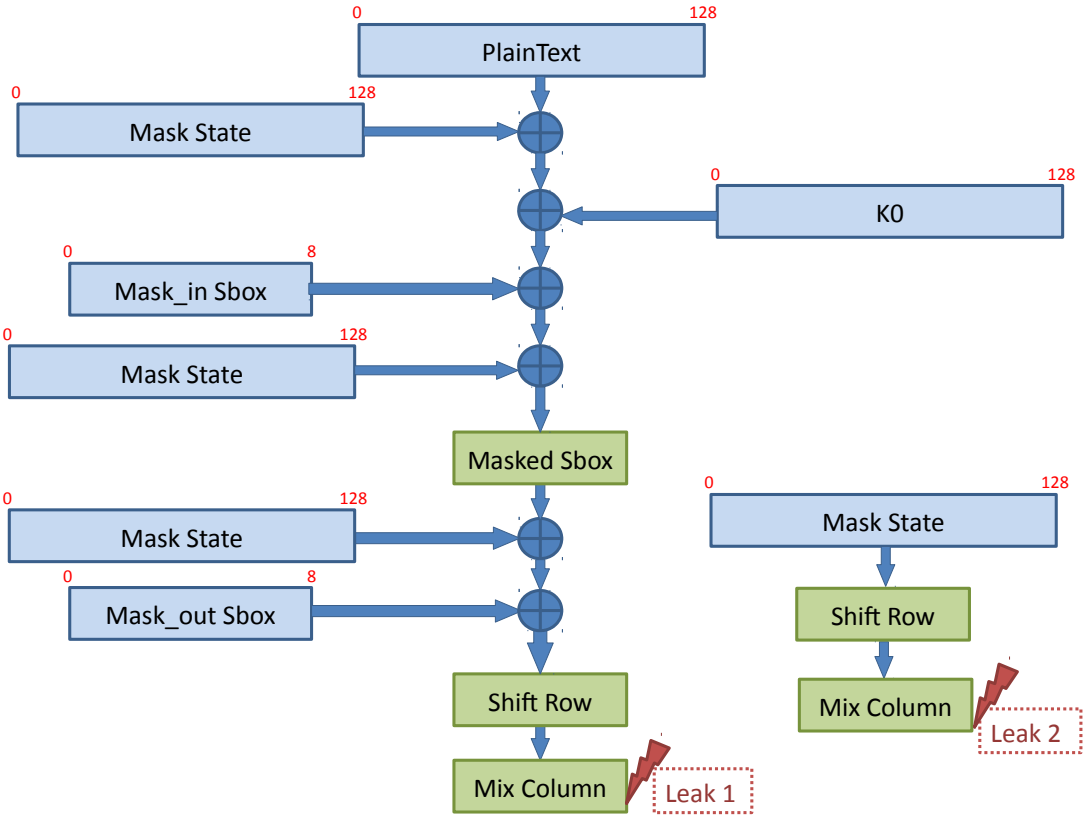


Figure 2.9: Masking scheme of the Plebe1 implementation.

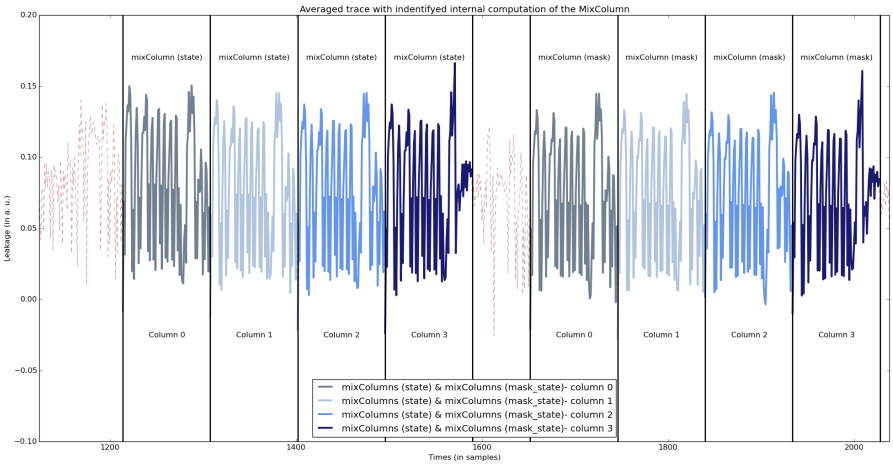


Figure 2.10: Side-Channel (SC) identification of the mixColumns computations.

where $\texttt{tmp}_b$ = S-box $(T_b^Q \oplus k)$, and . the Galois product, and with the following distinguisher step:

$$\{\mathcal{D}(X^{D',Q}_{\lfloor b/4 \rfloor}, Y^Q_{k,b})\}_{k<256, b<16}$$

2.5 Contributions

Dimension reduction. It is recognized that a sensitive data leaks over the time, which is manifested by the acquisition of few leaking samples. In this context, we proposed an innovative data reduction approach. The data reduction was usually viewed as a preprocessing. We demonstrate, in the context of univariate model, that the optimal distinguisher directly embed the dimension reduction. Furthermore, we generalize this new paradigm to multivariate models.

Leakage characterization and exploitation. We proposed in this thesis an accurate evaluation of the leakage. The exactness of the characterization leads us to mount stochastic attacks exploiting the whole leakage. This approach permits to take advantage of the huge variety of existing leakage: the data does not only leak through the perpetually used HW model. Complex models exist, each bit but also each combination of bits can leak through its own model. Furthermore, we can easily imagine a model that makes the HW model counterproductive, for examples if the focused byte leaks through the following model:

$$X = \sum_{i=0}^{7} (-1)^i Y^i + N$$

Software Analysis. We present a software analysis based on data collected via a debugger, we choose GNU Debugger (GDB) for its capability and its usability but it can be replaced by another data providers. One novelty of this contribution lies in the diversity of the collected data. In fact, we give a methodology that detect leakage from any kind of data manipulated during a code execution. Thus, we bring to light leakage ensue from flags registers. This result shows that leakage could follow from any kind of storage. Additionally, we proposed a pioneering feature of resynchronization. Our proposed realignment algorithm takes advantage of the auxiliary information provided by the program counter. Once traces are resynchronized, we are sure to analyze data

that results from the same operation at the same iteration. Moreover, the realignment process is fast and efficient as it only needs one read of the program counters and is based on accumulation.

2.6 The Thesis

The thesis is divided in three main chapters.

Less is more. As we can notice in the *Fig. 2.6*, whichever the complexity of the model, sensitive data leaks over the time in some samples. In this first *Chap. 3* we answer to the following question:

How can we optimally capture multidimensional leakage as one single compressed sample without loose of efficiency?

This study shows that optimal attacks remain optimal after a first pass of dimension reduction, which takes the form of a linear projection of the samples. We then investigate the state-of-the-art dimensionality reduction techniques, and find that asymptotically, the optimal strategy coincides with the Linear Discriminant Analysis (LDA). The two main objectives of the dimension reduction are the reduction of computational complexity and the exploitation of the leakage in its entirety. This *Chap. 3* gives rise to a first publication at CHES-2015 [13].

Multivariate Leakages and Multiple Models. Once a solution has been found to the problem of multivariate leakage, the question that naturally follows is:

How can we optimally capture multidimensional leakage exploiting multivariate models?

In the state-of-the-art, these two issues have two independent solutions: on the one hand, dimensionality reduction can cope with multivariate leakage; on the other hand, stochastic approach can cope with multiple models. In the *Chap. 4*, we combine both solutions to derive closed-form expressions of the resulting *optimal* distinguisher, in all situations where the model can be either profiled offline or regressed online. We recover known results for mono and bivariate models (including CPA), and investigate novel distinguishers for multiple models with $S \geq 2$. In addition, following ideas from the AsiaCrypt'2013 paper [49], we provide fast computation algorithms in which the traces are accumulated prior to computing the distinguisher values. This *Chap. 4* gave rise to

a first publication at PROOFS-2016 [14], to a poster at CHES-2016 and to an extended version in the journal of cryptographic engineering [15].

Binary Data Analysis for Source Code Leakage Assessment. In this *Chap. 5*, we keep increasing dimension. We treat a case where models Y are $S \times Q$ matrices and the collected data X is a $D \times R \times Q$ matrix. We meet this problematic collecting data provided by GDB. In this setting, we proposed an answer to the question:

How can we identified and characterize leaking information exploiting 3*-dimensional "software traces"?*

We proposed a methodology of data collection and analysis to identify potential leakage from any software implementation. We introduce a multivariate leakage analysis to extract the leaking points from the data. Then, we leverage on GDB to keep track of the execution context to map the identified leakage to the source code. We succeed to overcome two main difficulties: the misalignment and the multiplicity of leaking resources. Finally, we show how we can identify leakage in the source code applying our solution to a WBC implementation. This *Chap. 5* gave rise to a first publication at DTIS-2018 [10] and another one at SecITC-2018 [32].

CHAPTER 3

Less Is More

Contents

3.1 Contributions

In this chapter, we tackle the problem of dimensionality reduction from a theoretical viewpoint. Provided that the attacker has full knowledge of the leakage model, we find that "less is more": the advantages of dimensionality reduction can come with no impact on the attack success probability, while improving computational speed.

We derive that the optimal dimensionality reduction process consists in a linear

combination of samples, which we explicit as a projection on a specific one-dimensional space. For white noise, it turns out that the improved Signal-to-Noise Ratio (SNR) after projection is simply the sum of the SNR at the various samples before projection.

Finally, we show that the optimal dimensionality reduction technique asymptotically matches the Linear Discriminant Analysis (LDA) preprocessing. We find that LDA generally outperforms Principal Component Analysis (PCA) for which the SNR increases to a lesser extend than LDA, except in the case of white homoscedastic noise where PCA and LDA become equivalent.

We also validate in practice those results on the DPA contest V2 [73] traces.

3.2 Review of the State-of-the-Art.

Side-Channel Analysis (SCA) exploits leakages from devices. Embedded systems are targets of choice for such attacks. Typical leakages are captured by instruments such as oscilloscopes, which sample power or electromagnetic traces. The resulting leaked information about sensitive variables is spread over time.

In practice, two different attack strategies coexist. On the one hand, the various leaked samples can be considered individually—this is typical of non-profiled attacks such as Correlation Power Analysis [11]. On the other hand, profiled attacks characterize the leakage in a preliminary phase. An efficient leakage modelization should then involve a multidimensional probabilistic representation [18].

The large number of samples to feed into the model has always been a problematic issue for multidimensional SCA. One solution is to use techniques to select Points-of-Interest (PoI). Most of them, such as Sum-Of-Square Differences (SOSD) and t-test (SOST) [34], are *ad hoc* in that they result from a criterion which is independent from the attacker's key extraction objective. Recent criteria, such as leakage maximization by sensitive value [5], avoid this problem. Other formal criteria, related to non-profiled attacks, have also been proposed [40, 58].

Therefore, there seems to be a converging effort, in both non-profiled and profiled

attacks, to reduce the dimensionality of multidimensional measurements. This desirable property of dimensionality reduction achieves several goals simultaneously:

- it simplifies the Side-Channel (SC) problem (to a single multivariate probability density function);
- it concentrates the information (to distinguish using fewer traces); and
- it improves computational speed.

It can be argued, however, that like every preprocessing technique, dimensionality reduction would lose information.

Dimensionality reduction is part and parcel of profiled attacks. The seminal paper on Template Attack (TA) [18] is motivated by keeping covariance matrices involved in the training phase sufficiently well conditioned. Manual selection of relevant leaking points was discussed in [59] as educated guesses. Several automated techniques were proposed, such as SOSD and T-test (SOST) [34], and also wavelet transforms [27]. Several related metrics were proposed for leakage detection. The ANalysis Of VAriance (ANOVA) or F-test is a ratio between the explained variance and the total variance—see e.g. [25, 57] and [8] where it is named Normalized Inter-Class Variance (NICV). Also used for Linear Regression Analysis (LRA), it is known as the coefficient of determination, denoted by the symbol R^2. It is employed in the context of SCA in [72] as multivariate regression analysis in the presence of white noise, and in [68], where it is used as a distinguisher and as a linearity metric.

PCA has been used to compact traces in [7] and templates in [5]. The eigenvalues of PCA can be viewed as a security metric [37] or even as a distinguisher [69]. This technique is particularly attractive as it can be easily and accurately computed with no divisions involved. It is advocated in [44] that PCA aims at maximizing the inter-class variance, yet it is also important to take the intra-class variance into account. For this reason, LDA has been promoted as an improved alternative. Empirical comparisons were investigated in [62, 70, 71]. Unfortunately, despite some differences in terms of qualitative efficiency, there is no clear rationale to prefer one method over the other. In fact, it is unclear which of the intrinsic virtue of statistical tools, their implementation, or the dataset is actually responsible for the performance of dimensionality reduction.

Other works attempted to consider different objective functions. In [58], the cor-

rect key correlation is taken as the objective to be maximized. A similar goal is pursued in [38, 39, 40, 41]. Still other dimensionality reduction techniques exist, such as quadratic discriminant analysis, but have not been studied in the SC literature. similar questions have also been raised in the presence of masking countermeasures [12, 30, 63].

3.3 Theoretical Solution in the Presence of Gaussian Noise

3.3.1 Optimal Attack

We focus on the optimal attack as part of our scientific approach to the problem. It is always possible that for some peculiar reason a suboptimal attack actually performs better in the presence of dimensionality reduction. But by the data processing theorem [23] any preprocessing like dimensionality reduction can only decrease information about the secret, and, therefore, degrade performance of the optimal attack. As a result, it does make sense to minimize the impact of dimensionality reduction on the success rate for this optimal attack so as not to be biased by performance loss or gain due to other factors.

The optimal attack, also known as the TA [18], consists in applying the *maximum likelihood* principle [42]. Having collected Q traces of dimensionality D in a matrix $X^{D,Q}$, where each trace $X^{D,q}$ corresponds to a known plaintext T_q, the best key guess that maximizes the probability of success is given by

$$\mathcal{D}(X^{D,Q}, T^Q) = \underset{k}{\operatorname{argmax}}\ p(X^{D,Q}|T^Q, k^\star = k) \tag{3.1}$$

$$= \underset{k}{\operatorname{argmax}}\ p_{N^{D,Q}}(X^{D,Q} - \alpha^D Y^Q(k)) \tag{3.2}$$

$$= \underset{k}{\operatorname{argmax}} \prod_{q=1}^{Q} p_{N_q^D}(X_q^D - \alpha^D Y_q(k)) \tag{3.3}$$

where

$$p_{N_q^D}(z^D) = \frac{1}{\sqrt{(2\pi)^D |\det \Sigma|}} \exp\Big(-\frac{1}{2}(z^D)^\intercal \Sigma^{-1} z^D\Big). \tag{3.4}$$

We have used the independence of the queries in *Eq. 3.3* and the assumption that at each query, the noise distribution is the same in *Eq. 3.4*.

Notice that, the optimal attack can as well be a Simple Power Analysis (SPA) (if $Q = 1$) or a Differential Power Analysis (DPA) (if $Q > 1$), using the terminology from [46]. Still, in the sequel, we focus on attacks which require many traces ($Q \gg 1$).

3.3.2 Optimal Dimensionality Reduction

We state our main result in the following Theorem 3.1:

Theorem 3.1. *The optimal attack on the multivariate traces $X^{D,Q}$ is equivalent to the optimal attack on the univariate traces $\tilde{X}^Q$, obtained from $X^{D,Q}$ by the formula:*

$$\tilde{X}_q = \frac{(\alpha^D)^\intercal \Sigma^{-1} X_q^D}{(\alpha^D)^\intercal \Sigma^{-1} \alpha^D} \qquad (q = 1, \ldots, Q). \tag{3.5}$$

Proof. By taking the logarithm of the expression to be maximized in *Eqs. 3.1*–3.4, the optimal distinguisher $\mathcal{D}(X^{D,Q}, T^Q)$ rewrites

$$\mathcal{D}(X^{D,Q}, T^Q) = \underset{k}{\operatorname{argmin}} \sum_{q=1}^{Q} \left(X_q^D - \alpha^D Y_q(k)\right)^\intercal \Sigma^{-1} \left(X_q^D - \alpha^D Y_q(k)\right). \tag{3.6}$$

For each trace index q, the terms in the sum expand to

$$\begin{aligned} &\underbrace{(X_q^D)^\intercal \Sigma^{-1} X_q^D}_{\text{cst. } C \text{ independent of } k} - 2(\alpha^D)^\intercal Y_q(k) \Sigma^{-1} X_q^D + (Y_q(k))^2 (\alpha^D)^\intercal \Sigma^{-1} \alpha^D \\ &\qquad = C - 2Y_q(k)\left[(\alpha^D)^\intercal \Sigma^{-1} X_q^D\right] + (Y_q(k))^2 \left[(\alpha^D)^\intercal \Sigma^{-1} \alpha^D\right] \\ &\qquad = \left[(\alpha^D)^\intercal \Sigma^{-1} \alpha^D\right] \left(Y_q(k) - \frac{(\alpha^D)^\intercal \Sigma^{-1} X_q^D}{(\alpha^D)^\intercal \Sigma^{-1} \alpha^D}\right)^2 + C'. \end{aligned}$$

The latter division is valid since Σ is positive definite and α^D is a nonzero vector. Therefore,

$$\begin{aligned} \mathcal{D}(X^{D,Q}, T^Q) &= \underset{k}{\operatorname{argmin}} \sum_{q=1}^{Q} \left(Y_q(k) - \frac{(\alpha^D)^\intercal \Sigma^{-1} X_q^D}{(\alpha^D)^\intercal \Sigma^{-1} \alpha^D}\right)^2 \left[(\alpha^D)^\intercal \Sigma^{-1} \alpha^D\right] \\ &= \underset{k}{\operatorname{argmin}} \sum_{q=1}^{Q} \frac{\left(\tilde{X}_q - Y_q(k)\right)^2}{\tilde{\sigma}^2}, \end{aligned} \tag{3.7}$$

where

$$\begin{cases} \tilde{X}_q &= \dfrac{(\alpha^D)^\intercal \Sigma^{-1} X_q^D}{(\alpha^D)^\intercal \Sigma^{-1} \alpha^D}, \\ \tilde{\sigma} &= \left((\alpha^D)^\intercal \Sigma^{-1} \alpha^D\right)^{-1/2}. \end{cases} \tag{3.8}$$

We have shown that *Eq. 3.6* and *Eq. 3.7* are equivalent expressions for the same optimal distinguisher, computed either:

- on multivariate traces X_q^D, with a noise covariance matrix Σ, or:
- on univariate (i.e., scalar) traces $\tilde{X}_q$, with scalar noise of variance $\tilde{\sigma}^2$.

□

Theorem 3.1 shows that in fact, the optimal attack already integrates an optimal dimensionality reduction. The maximal success rate is not altered.

Definition 3.1 (Projection vector)**.** *Let V^D be a column of D elements. We call the projection of an acquisition campaign $X^{D,Q}$ on V^D the new mono-sample traces $(V^D)^\intercal X^{D,Q}$. That is, every trace X_q^D $(1 \leq q \leq Q)$ of the initial campaign is summarized as one sample $(V^D)^\intercal X_q^D = \langle V^D \mid X_q^D \rangle$.*

Based on this definition, *Theorem 3.1* can be interpreted as follows.

Corollary 3.1. *The optimal dimensionality reduction is made by a linear combination of the samples where each multivariate trace is projected on the vector $V^D = \dfrac{\Sigma^{-1}\alpha^D}{(\alpha^D)^\intercal \Sigma^{-1} \alpha^D}$, of size $D \times 1$.*

Proof. By *Therorem 3.1*,

$$\underbrace{\tilde{X}^Q}_{1 \times Q \text{ matrix}} = \underbrace{\frac{(\alpha^D)^\intercal \Sigma^{-1}}{(\alpha^D)^\intercal \Sigma^{-1} \alpha^D}}_{1 \times D \text{ matrix } (V^D)^\intercal} \underbrace{X^{D,Q}}_{D \times Q \text{ matrix}}.$$

□

In addition, after this projection, the leakage becomes scalar and can be characterized by a SNR as shown as follow:

Corollary 3.2. *After optimal dimensionality reduction, the SNR is given by*

$$\frac{1}{\tilde{\sigma}^2} = (\alpha^D)^\intercal \Sigma^{-1} \alpha^D.$$

Proof. This is in line with *Eq. 3.7*. The random leakage $X^{D,Q}$ is protected onto V^D to yield $\tilde{X}_q = Y_q(k) + \tilde{N}$ $(q = 1, \ldots, Q)$ where $\tilde{N}$ is an additive Additive White Gaussian Noise (AWGN) distributed as $\mathcal{N}(0, ((\alpha^D)^\intercal \Sigma^{-1} \alpha^D)^{-1})$. Recall that the variance of the leakage model has been assumed normalized $= 1$. Therefore, SNR equals

$$\frac{\mathrm{var}(Y_q(k))}{\mathrm{var}(\tilde{N})} = \frac{1}{((\alpha^D)^\intercal \Sigma^{-1} \alpha^D)^{-1}} = (\alpha^D)^\intercal \Sigma^{-1} \alpha^D \ .$$

□

The SNR is an interesting metric on its own, because it quantifies how much the signal has been concentrated (its power increased) for a given noise level. Furthermore, the SNR directly relates to the success rate of optimal attacks [33].

Comment 3.1. *Reminder:*

- $(1+x)^n = \sum_{h=0}^{n} x^h \binom{h}{n}$, *evaluated in* $x = 1$ *yields:* $\sum_{h=0}^{n} \binom{h}{n} = 2^n$.
- *Thus:* $\frac{\partial}{\partial x}(1+x)^n = n(1+x)^{n-1} = \sum_{h=1}^{n} h x^{h-1} \binom{h}{n}$, *evaluated in* $x = 1$ *yields:* $\sum_{h=1}^{n} h \binom{h}{n} = \sum_{h=0}^{n} h \binom{h}{n} = n2^{n-1}$.
- *Thus:* $\frac{\partial^2}{\partial^2 x}(1+x)^n = n(n-1)(1+x)^{n-2} = \sum_{h=2}^{n} h(h-1) x^{h-2} \binom{h}{n}$, *evaluated in* $x = 1$ *yields:* $\sum_{h=2}^{n} h(h-1) \binom{h}{n} = \sum_{h=0}^{n} h(h-1) \binom{h}{n} = n(n-1)2^{n-2}$.

Thus $\sum_{h=0}^{n} h^2 \binom{h}{n} = \sum_{h=0}^{n} (h(h-1)+h) \binom{h}{n} = n(n-1)2^{n-2} + n2^{n-1} = n(n+1)2^{n-2}$, *and:*

$$Var(w_H(T) - \frac{n}{2}) = \mathbb{E}(w_H(T) - \frac{n}{2})^2 = \frac{1}{2^n} \sum_{h=0}^{n} h^2 \binom{h}{n} - \left(\frac{n}{2}\right)^2 = \frac{n(n+1)}{4} - \frac{n^2}{4} = \frac{n}{4}.$$

3.3.3 Discussion

It is interesting to note that the optimal dimensionality reduction does not depend on the actual distribution of $Y^D(k)$, the deterministic part of the leakage model. This means that irrespective of the leakage function ϕ, the best dimensionality reduction depends only on signal weights α^D and on noise covariance Σ.

Similarly, the optimal dimensionality reduction does not depend on the *confusion coefficient* of the leakage model [33]: for identical weight and noise distribution, the optimal linear combination of leakages is the same whether a eXclusive OR (XOR) or a substitution box operation is targeted.

3.4 Noise Distributions

3.4.1 White Noise

One interesting situation is when the noise samples are uncorrelated (see for instance [72] for an experimental setup). The covariance matrix Σ is diagonal:

$$\Sigma = \begin{pmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_D^2 \end{pmatrix}.$$

Proposition 3.1. *For white noise, the optimal dimensionality reduction takes the form:*

$$\tilde{X}_q = \frac{\sum_{d=1}^{D} \frac{\alpha_d}{\sigma_d^2} X_{d,q}}{\sum_{d=1}^{D} \frac{\alpha_d^2}{\sigma_d^2}} \qquad (q = 1, \ldots, Q) \tag{3.9}$$

Proof. Apply *Theorem 3.1*, where Σ^{-1} is diagonal with diagonal entries $1/\sigma_d^2$. □

Let $\mathrm{SNR}_d = \alpha_d^2/\sigma_d^2$ be the initial SNR at the d^{th} sample *before* dimensionality reduction.

Proposition 3.2. *For white noise, the equivalent SNR* after *optimal dimensionality reduction is given by the sum*

$$\widetilde{SNR} = \sum_{d=1}^{D} SNR_d. \tag{3.10}$$

Proof. By *Corollary 3.2*, $\widetilde{\mathrm{SNR}} = (\alpha^D)^\intercal \Sigma^{-1} \alpha^D = \sum_{d=1}^{D} \frac{\alpha_d^2}{\sigma_d^2} = \sum_{d=1}^{D} \mathrm{SNR}_d$. □

Thus, combining independent multidimensional samples within one trace increases the SNR as if those samples were captured in D independent traces. In this case having Q traces of D samples each is simply the same as having $Q \times D$ independent univariate traces.

3.4.2 Correlated Autoregressive Noise

A more general situation is when the samples are correlated like an autoregressive process. More precisely, assume that all samples share the same noise distribution of variance σ^2, and that two consecutive noise samples have correlation factor equal to $\rho \in]-1, +1[$. The correlation factors ρ typically models an autoregressive low-pass filtering of the acquisition setup (see *Sec. 3.6* for a real-world example). The noise covariance matrix takes the Toeplitz form:

$$\Sigma = \sigma^2 \begin{pmatrix} 1 & \rho & \rho^2 & \rho^3 & \cdots & \rho^{D-2} & \rho^{D-1} \\ \rho & 1 & \rho & \rho^2 & \cdots & \rho^{D-3} & \rho^{D-2} \\ \rho^2 & \rho & 1 & \rho & \cdots & \rho^{D-4} & \rho^{D-3} \\ \rho^3 & \rho^2 & \rho & 1 & \cdots & \rho^{D-5} & \rho^{D-4} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \rho^{D-2} & \rho^{D-3} & \rho^{D-4} & \rho^{D-5} & \cdots & 1 & \rho \\ \rho^{D-1} & \rho^{D-2} & \rho^{D-3} & \rho^{D-4} & \cdots & \rho & 1 \end{pmatrix} = \left(\sigma^2 \rho^{|d-d'|}\right)_{1 \leq d,d' \leq D}.$$

We emphasize that $|\rho|$ is strictly smaller than one in keeping with the assumption that Σ be positive definite. When $\rho = 0$, the noise becomes white as in the preceding subsection.

Proposition 3.3. *For autoregressive noise, the optimal dimensionality reduction takes the form:*

$$\tilde{X}_q = \tfrac{1}{\sigma^2(1-\rho^2)}\Big[(\alpha_1 - \rho\alpha_2)X_{q,1} + \sum_{d=2}^{D-1}((1+\rho^2)\alpha_d - \rho(\alpha_{d-1}+\alpha_{d+1}))X_{d,q} + (\alpha_D - \rho\alpha_{D-1})X_q^D\Big]. \quad (3.11)$$

Proof. It can easily be checked that Σ^{-1} is tridiagonal, we can do that here, without risk of collision

$$\Sigma^{-1} = \frac{1}{\sigma^2(1-\rho^2)}\begin{pmatrix} 1 & -\rho & 0 & 0 & \cdots & 0 & 0 \\ -\rho & 1+\rho^2 & -\rho & 0 & \cdots & 0 & 0 \\ 0 & -\rho & 1+\rho^2 & -\rho & \cdots & 0 & 0 \\ 0 & 0 & -\rho & 1+\rho^2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1+\rho^2 & -\rho \\ 0 & 0 & 0 & 0 & \cdots & -\rho & 1 \end{pmatrix}.$$

Then apply *Theorem 3.1*:

$$\tilde{X}_q = \tfrac{1}{\sigma^2(1-\rho^2)}\begin{pmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_{D-1} & \alpha_D \end{pmatrix} \times \begin{pmatrix} 1 & -\rho & \cdots & 0 & 0 \\ -\rho & 1+\rho^2 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1+\rho^2 & -\rho \\ 0 & 0 & \cdots & -\rho & 1 \end{pmatrix}\begin{pmatrix} X_{q,1} \\ X_{q,2} \\ \vdots \\ X_{q,D-1} \\ X_{q,D} \end{pmatrix}$$

and expand. □

Notice that in the optimal dimensionality reduction, each leakage sample $X_{d,q}$ is not only weighted by its corresponding α_d but also by its two neighbor weights $\alpha_{d\pm1}$, provided the latter exist.

Proposition 3.4. *For autoregressive noise, the equivalent SNR after optimal dimensionality reduction is given by*

$$\widetilde{SNR} = \frac{1}{\sigma^2(1-\rho^2)}\big[\alpha_1^2 + (1+\rho^2)\sum_{d=2}^{D-1}\alpha_d^2 + \alpha_D^2 - 2\rho\sum_{d=1}^{D-1}\alpha_d\alpha_{d+1}\big]. \tag{3.12}$$

Proof. Apply *Corollary 3.2*:

$$\widetilde{\mathrm{SNR}} = \frac{1}{\sigma^2(1-\rho^2)}\begin{pmatrix}\alpha_1 & \alpha_2 & \cdots & \alpha_{D-1} & \alpha_D\end{pmatrix} \times \begin{pmatrix} 1 & -\rho & \cdots & 0 & 0 \\ -\rho & 1+\rho^2 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1+\rho^2 & -\rho \\ 0 & 0 & \cdots & -\rho & 1 \end{pmatrix}\begin{pmatrix}\alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_{D-1} \\ \alpha_D\end{pmatrix}$$

and expand. □

Corollary 3.3. *For equal weights $\alpha_1 = \cdots = \alpha_D = \alpha$, i.e., when initial $SNR_1 = \cdots = SNR_D = SNR$ are the same, one has*

$$\widetilde{SNR} = SNR \times \frac{D(1-\rho)+2\rho}{1+\rho}. \tag{3.13}$$

Proof. *Proposition 3.4* reduces to

$$\begin{aligned}\widetilde{\mathrm{SNR}} &= \frac{\alpha^2}{\sigma^2(1-\rho^2)}\big(2+(D-2)(1+\rho^2)-2\rho(D-1)\big) \\ &= \frac{\alpha^2}{\sigma^2(1-\rho)(1+\rho)}\left((1-\rho)(D-\rho(D-2))\right) \\ &= \frac{\alpha^2}{\sigma^2}\frac{1}{1+\rho}\left(D-\rho(D-2)\right) = \mathrm{SNR}\times\frac{D(1-\rho)+2\rho}{1+\rho}.\end{aligned}$$

□

In other words, optimal dimensionality reduction has the effect of multiplying the univariate SNR by the factor $\frac{D-\rho(D-2)}{1+\rho}$. This gain factor is of course equal to 1 for

dimension $D = 1$, but becomes strictly greater than 1 for larger dimensions, since $\frac{D-\rho(D-2)}{1+\rho} > \frac{D-(D-2)}{2} = 1$ where we have used that $\rho > -1$ or $\frac{1}{1+\rho} > \frac{1}{2}$.

For very small values of correlation ρ, Taylor expansion about $\rho = 0$ gives $\frac{D-\rho(D-2)}{1+\rho} = D - 2(D-1)\rho + \mathcal{O}(\rho^2)$. The SNR gain is equal to the dimension D at first order, which is consistent with Proposition 3.2. In addition, that gain is never greater than D, since $\frac{D(1-\rho)+2\rho}{1+\rho} \leq \frac{D(1-\rho)+2D\rho}{1+\rho} = D$. Therefore, when $\mathrm{SNR}_1 = \ldots = \mathrm{SNR}_D$, nonzero values of correlation ρ decrease the efficiency of dimensionality reduction, the most favorable situation being the case of white noise samples.

3.5 Comparison with PCA and LDA

When the attacker does not precisely know the model given by *Eq. 2.1*, the optimal dimensionality reduction cannot be applied directly. In this section, we analyze theoretically two well-known engineering solutions to reduce the dimensionality: PCA and LDA. Both techniques are based on eigen decompositions.

3.5.1 Principal Component Analysis (PCA)

PCA aims at identifying directions in the *centered* data set $M^{D,Q} = (M_{d,q})_{d,q}$ defined by

$$M_{d,q} = X_{d,q} - \frac{1}{Q}\sum_{q'=1}^{Q} X_{d,q'} \qquad (1 \leq q \leq Q, 1 \leq d \leq D). \tag{3.14}$$

The directions of PCA are the eigenvectors of $M^{D,Q}(M^{D,Q})^\intercal$.

Proposition 3.5. *Asymptotically as* $Q \longrightarrow +\infty$,

$$\frac{1}{Q}M^{D,Q}(M^{D,Q})^\intercal \longrightarrow \alpha^D(\alpha^D)^\intercal + \Sigma. \tag{3.15}$$

Proof. By the law of large numbers,

$$\frac{1}{Q}\sum_{q=1}^{Q} M_{d,q}M_{d',q} \longrightarrow \mathrm{cov}(M_{d,q}, M_{d',q})$$

almost surely, where the covariance term can be computed as: $\mathrm{cov}(M_{d,q}, M_{d',q}) = \mathrm{cov}(\alpha_d Y_q + N_{d,q}, \alpha_{d'} Y_q + N_{d',q})$. When expanding this expression, cross terms disappear by independence of Y^Q and $N^{D,Q}$. There remains:

$$\mathrm{cov}(M_{d,q}, M_{d',q}) = \alpha_d \alpha_{d'} + \Sigma_{d,d'}$$

where we have used the hypothesis that Y_q has unit variance. □

The classical PCA has the drawback that $M^{D,Q}(M^{D,Q})^\mathsf{T}$ depends both on the *signal* and on the *noise*. *Inter-class PCA* has been introduced in [5]. The matrix $M^{D,Q}$ used in the PCA is traded for a more simple matrix $Z^{D,\#Y}$, where each column, indexed by y, is the centered column $\dfrac{1}{\sum\limits_{\substack{1\le q\le Q\\ Y_q=y}} 1}\sum\limits_{\substack{1\le q\le Q\\ Y_q=y}} X_q^D$. One advantage of this method is that it explicitly takes into account the sensitive variable Y.

It can be easily checked, that, asymptotically, each column Z_y^D tends to $\alpha^D Y$ when $Q \longrightarrow +\infty$. Therefore, $Z^{D,\#Y}(Z^{D,\#Y})^\mathsf{T}$ tends to a $D \times D$ matrix proportional to $\alpha^D(\alpha^D)^\mathsf{T}$. Here, the noise has been averaged away in each class of Y, which is a second advantage. Therefore, in the sequel, shall refer to the inter-class PCA of [5] simply as PCA.

We have the following spectral characterization of the asymptotic PCA:

Proposition 3.6. *Asymptotically, PCA has only one principal direction, namely the vector* α^D.

Proof. By *Proposition 3.5*, the PCA matrix tends asymptotically to $\alpha^D(\alpha^D)^\mathsf{T}$. This $D \times D$ matrix has rank one, because all its columns are multiple of α^D. Since

$$(\alpha^D(\alpha^D)^\mathsf{T})\alpha^D = \alpha^D((\alpha^D)^\mathsf{T}\alpha^D) = \left\|\alpha^D\right\|_2^2 \times \alpha^D,$$

α^D is the eigenvector with corresponding nonzero eigenvalue $= \|\alpha^D\|_2^2$. □

Notice that the uniqueness of the eigenvector for PCA holds in our model *Eq. 2.1*. However, *Proposition 3.6* would not hold if e.g., the noise were correlated to the signal.

Remark 3.1. *The classical PCA has the same eigenvector* α^D *if the noise is* isotropic*, i.e., white and of same variance in every dimension.*

The paper [5] presents an optimization procedure to find the eigenelements.

Proposition 3.7. *The asymptotic SNR after projection using PCA is equal to* $\frac{\|\alpha^D\|_2^4}{(\alpha^D)^\mathsf{T}\Sigma\alpha^D}$.

Proof. After projection on the (asymptotic) eigenvector α^D, the leakage becomes: $(\alpha^D)^\mathsf{T}\alpha^D Y_q(k^\star) + (\alpha^D)^\mathsf{T} N_q^D$. The projected signal is $((\alpha^D)^\mathsf{T}\alpha^D)Y_q(k^\star)$. The projected noise is $(\alpha^D)^\mathsf{T} N_q^D$, which remains centered. Its variance is equal to the expectation of its square:

$$\begin{aligned}\operatorname{var}((\alpha^D)^\mathsf{T} N_q^D) &= \mathbb{E}\left((\alpha^D)^\mathsf{T} N_q^D\right)^2 = \mathbb{E}\left((\alpha^D)^\mathsf{T} N_q^D (N_q^D)^\mathsf{T}\alpha^D\right) \\ &= (\alpha^D)^\mathsf{T}\mathbb{E}\left(N_q^D (N_q^D)^\mathsf{T}\right)\alpha^D = (\alpha^D)^\mathsf{T}\Sigma\alpha^D.\end{aligned}$$

Therefore,

$$\mathrm{SNR}_{\mathrm{PCA}} = \frac{\operatorname{var}(((\alpha^D)^\mathsf{T}\alpha^D)Y_q(k^\star))}{\operatorname{var}((\alpha^D)^\mathsf{T} N_q^D)} = \frac{\operatorname{var}(\|\alpha^D\|_2^2 Y_q(k^\star))}{(\alpha^D)^\mathsf{T}\Sigma\alpha^D} = \frac{\|\alpha^D\|_2^4}{(\alpha^D)^\mathsf{T}\Sigma\alpha^D}.$$

□

Example 3.1. *For white noise (Subsec. 3.4.1)*

$$SNR_{PCA} = \frac{\left(\sum_{d=1}^{D}\alpha_d^2\right)^2}{\sum_{d=1}^{D}\alpha_d^2\sigma_d^2}. \tag{3.16}$$

Example 3.2. *For autoregressive noise (Subsec. 3.4.2)*

$$SNR_{PCA} = \frac{\sum_{d=1}^{D} \alpha_d^2}{\sigma^2} \frac{1}{1 + \frac{2}{\sum_{d=1}^{D} \alpha_d^2} \sum_{d=1}^{D-1} \rho^d \sum_{d'=1}^{D-d} \alpha_{d'} \alpha_{d'+d}}. \tag{3.17}$$

We can now compare the performance of the asymptotic PCA to the optimal dimensionality reduction.

Theorem 3.2. *The SNR of the asymptotic PCA is smaller than the SNR of the optimal dimensionality reduction.*

Proof. By assumption the noise covariance matrix is symmetric positive definite, hence there exists a matrix $\Sigma^{1/2}$, which is such that $\Sigma^{1/2}\Sigma^{1/2} = \Sigma$. By Cauchy-Schwarz inequality,

$$\left(\langle \Sigma^{-1/2}\alpha^D \mid \Sigma^{1/2}\alpha^D \rangle\right)^2 \leq \left\|\Sigma^{-1/2}\alpha^D\right\|_2^2 \cdot \left\|\Sigma^{1/2}\alpha^D\right\|_2^2.$$

Therefore, $\mathrm{SNR}_{\mathrm{PCA}} = \frac{((\alpha^D)^\mathsf{T}\alpha^D)^2}{(\alpha^D)^\mathsf{T}\Sigma\alpha^D} \leq (\alpha^D)^\mathsf{T}\Sigma^{-1}\alpha^D = \widetilde{SNR}$. □

Corollary 3.4. *The asymptotic PCA has the same SNR as the the optimal dimensionality reduction if and only if α^D is an eigenvector of Σ. In this case, both dimensionality reductions are equivalent.*

Proof. Equality holds in *Theorem 3.2* if and only if there exists a nonzero real number λ such that $\Sigma^{1/2}\alpha^D = \lambda\Sigma^{-1/2}\alpha^D$, i.e., $\Sigma\alpha^D = \lambda\alpha^D$, i.e., α^D is an eigenvector of Σ.

In this case, the optimal protection is on the vector $\Sigma^{-1}\alpha^D = \frac{1}{\lambda}\alpha^D$, which is proportional to the projection vector belonging to the asymptotic PCA. □

Remark 3.2. *Assume white noise (Subsec 3.4.1) where all values σ_d^2 $(1 \leq d \leq D)$ are different. Then, by Corollary 3.4, the asymptotic PCA is optimal only if $\alpha^D =$*

$(0, 0, \cdots, 0, 1, 0, \cdots, 0)$, *which we may consider unrealistic since only one sample out of* D *would leak secret information.*

In contrast, if $\sigma_1 = \cdots = \sigma_D = \sigma$, *the covariance matrix has only one eigenvalue, namely* $(1, 1, \cdots, 1)$, *which has multiplicity* D. *Thus, for white homoscedastic noise, PCA is asymptotically optimal if and only if* $\alpha_1 = \cdots \alpha_D = \alpha$, *that is, the SNR is the same for each sample.*

Still in the case of white noise, we can lower bound the SNR of the asymptotic PCA:

Lemma 3.1. *For white noise, the SNR of the asymptotic PCA is not less than the worst SNR among the samples, but can be strictly smaller than the higher SNR among the samples.*

Proof. We have

$$\sum_{d=1}^{D} \alpha_d^2 \sigma_d^2 = \sum_{d=1}^{D} \frac{\sigma_d^2}{\alpha_d^2} \alpha_d^4 \leq \left(\max_{d=1}^{D} \frac{\sigma_d^2}{\alpha_d^2} \right) \sum_{d=1}^{D} \alpha_d^4.$$

Since $\left(\max_{d=1}^{D} \frac{\sigma_d^2}{\alpha_d^2} \right)^{-1} = \min_{d=1}^{D} \frac{\alpha_d^2}{\sigma_d^2} = \min_{d=1}^{D} \mathrm{SNR}_d$, the expression of the SNR of the asymptotic PCA given by *Eq. 3.16* is such that

$$\mathrm{SNR}_{\mathrm{PCA}} = \frac{\left(\sum_{d=1}^{D} \alpha_d^2 \right)^2}{\sum_{d=1}^{D} \alpha_d^2 \sigma_d^2} \geq \frac{\left(\sum_{d=1}^{D} \alpha_d^2 \right)^2}{\sum_{d=1}^{D} \alpha_d^4} \min_{d=1}^{D} \mathrm{SNR}_d \geq \min_{d=1}^{D} \mathrm{SNR}_d \tag{3.18}$$

where we have used Cauchy-Schwarz inequality $\alpha_d^4 = \sum_{d=1}^{D} \alpha_d^2 \alpha_d^2 \leq \left(\sum_{d=1}^{D} \alpha_d^2 \right)^2$.

Conversely, we can give an example for which $\mathrm{SNR}_{\mathrm{PCA}} < \max_{d=1}^{D} \frac{\alpha_d^2}{\sigma_d^2}$. Take $D = 2$, $\alpha_1 = \alpha_2 = 1$, $\sigma_1 = 1$ and $\sigma_2 = 10$. Then $\mathrm{SNR}_{\mathrm{PCA}} = 4/(1 + 10^2) = 4/101$, which is strictly smaller than $\alpha_1^2/\sigma_1^2 = 1$. □

3.5.2 Linear Discriminant Analysis (LDA)

LDA has been introduced in SCA in [70]. With respect to inter-class PCA, it computes the eigenvectors of the matrix $S_w^{-1}S_b$, where:

- S_w is the *within-class scatter matrix*, asymptotically equal to Σ, and
- S_b is the *between-class scatter matrix*, equal to $\alpha^D(\alpha^D)^\mathsf{T}$.

We have the following spectral characterization of the asymptotic LDA:

Proposition 3.8. *Asymptotically, LDA has only one principal direction, namely the vector* $\Sigma^{-1}\alpha^D$.

Proof. The matrix $S_w^{-1}S_b = \Sigma^{-1}\alpha^D(\alpha^D)^\mathsf{T}$ has rank one. Indeed, $\alpha^D(\alpha^D)^\mathsf{T}$ has rank one, and multiplying by an invertible matrix (namely Σ^{-1}) keeps the rank unchanged. Since

$$(\Sigma^{-1}\alpha^D(\alpha^D)^\mathsf{T})\Sigma^{-1}\alpha^D = \Sigma^{-1}\alpha^D((\alpha^D)^\mathsf{T}\Sigma^{-1}\alpha^D) = \left((\alpha^D)^\mathsf{T}\Sigma^{-1}\alpha^D\right) \times \Sigma^{-1}\alpha^D,$$

$\Sigma^{-1}\alpha^D$ is the unique eigenvector with corresponding eigenvalue $(\alpha^D)^\mathsf{T}\Sigma^{-1}\alpha^D > 0$. This eigenvalue is equal to the SNR of the asymptotic LDA. □

By *Corollary 3.2*, the SNR of the asymptotic LDA is equal to the SNR of the optimal dimensionality reduction, denoted by $\widetilde{\text{SNR}}$. In fact, we have the following.

Theorem 3.3. *The asymptotic LDA computes exactly the optimal dimensionality reduction.*

Proof. Compare *Theorem 3.1* with *Proposition 3.8*: in both cases, the projection vector is collinear with $\Sigma^{-1}\alpha^D$. □

3.5.3 Numerical Comparison Between Asymptotic PCA and LDA

Numerical comparison between asymptotic PCA and LDA is given in *Fig. 3.1*(a) and (b), for $D = 6$ samples. The noise is chosen autoregressive, with $\sigma = 1$ and different values

for ρ (*Subsec. 3.4.2*). The vector α^D is chosen equal to $(1,1,1,1,1,1)^\mathsf{T}$ in *Fig. 3.1*(a) and to $\sqrt{6.0/6.4}\cdot(1.0,1.1,1.2,1.3,0.9,0.5)^\mathsf{T}$ in *Fig. 3.1*(b), such that $\widetilde{\mathrm{SNR}} = 6$ when $\rho = 0$. The SNR of the asymptotic LDA is that of the optimal dimensionality reduction (cf. *Corollary 3.2*), and that of the asymptotic PCA can be found in *Example 3.2*. The first case (*Fig. 3.1*(a)) fits the situation depicted in *Corollary 3.3*. The asymptotic PCA and LDA are almost similar. Besides, when $\rho \to 1^-$, both SNRs tend to 1 (recall *Eq. 3.17* and *Eq. 3.13*). But, when the SNR varies over the D samples (*Fig. 3.1*(b)), the asymptotic LDA can be significantly better than the asymptotic PCA. The sample-wise extremal SNRs ($\mathrm{SNR}_d = \alpha_d^2/\sigma^2$) are also represented: the SNR of the PCA can be smaller than the largest SNR, namely $\max_{1\le d\le D} \mathrm{SNR}_d$, (recall *Lemmas 3.1*), which is not the case of the SNR of the LDA. Actually, the SNR of LDA increases to infinity because $\widetilde{\mathrm{SNR}} \approx 0.164/(1-\rho)$ when $\rho \to 1^-$ (see *Eq 3.12*).

(a) Equal $\mathrm{SNR}_d = 1$, $1 \le d \le D$ (b) Varying SNR_d, $1 \le d \le D$

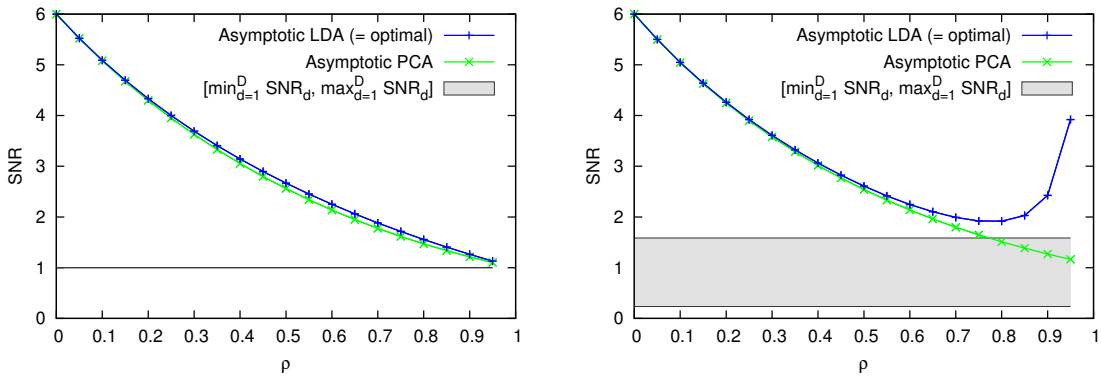


Figure 3.1: Comparison of the SNR of asymptotic LDA (optimal) and of asymptotic PCA

3.6 Practical Validations

In this section, we investigate real traces. Experiments are carried out on the DPA contest V2 [73] traces. One clock cycle lasts $D = 200$ samples. As traces are captured from a hardware implementation of an AES-128 , we consider the Hamming Distance (HD) leakage model (in accordance with most attacks reported on the analyzed device [20], namely a SASEBO-GII board with a Xilinx XC5VLX30 FPGA [66]). In the sequel, we focus on the HD between the byte 0 of the last round and that of the ciphertext.

That is, the function ϕ in *Eq. 2.1* is a normalized Hamming Weight (HW); precisely, $\phi : z \in \mathbb{F}_2{}^n \mapsto \frac{2}{\sqrt{n}}\left(\mathrm{HW}(z) - \frac{n}{2}\right)$, where $n = 8$, because AES-128 is a byte-oriented block cipher. In addition, we emphasize that our model (*Eq. 2.1*) is indeed suitable to leakage dimensionality reduction within one clock period.

3.6.1 Attack with a Precharacterization of the Model Parameters α^D and Σ

In order to characterize the model, we need to recover the column matrix α^D and the $D \times D$ covariance matrix Σ of the noise.

Proposition 3.9. *The parameters of the model Eq. 2.1 which minimize the fitting error are given by*

$$\hat{\alpha}^D = \frac{X^{D,Q}(Y^Q)^\mathsf{T}}{Y^Q(Y^Q)^\mathsf{T}}.$$

Proof. The goal (minimizing the fitting error) is similar to that of the optimal distinguisher, namely maximize the probability of $p_{N^{D,Q}}(X^{D,Q} - \alpha^D Y^Q)$ (*Eq. 3.3*). But in the context of characterization, the correct key is known. Therefore, we wish to minimize in α^D and Σ the following objective function:

$$\mathrm{objective}(\alpha^D, \Sigma) = \sum_{q=1}^{Q} \left\{ \left(X_q^D - \alpha^D y_q(k^\star)\right)^\mathsf{T} \Sigma^{-1} \left(X_q^D - \alpha^D y_q(k^\star)\right) \right\}, \tag{3.19}$$

which reminds of *Eq. 3.6* (except that now, the key $k = k^\star$ is known). We use the notation $(\hat{\alpha}^D, \hat{\Sigma}) = \underset{(\alpha^D, \Sigma)}{\mathrm{argmin}}\, \mathrm{objective}(\alpha^D, \Sigma)$.

We fix Σ and minimize only on α^D. The gradient of $\mathrm{objective}(\alpha^D, \Sigma)$ w.r.t. $(\alpha^D)^\mathsf{T}$ writes:

$$\frac{\partial}{\partial (\alpha^D)^\mathsf{T}} \mathrm{objective}(\alpha^D, \Sigma) = \sum_{q=1}^{Q} -2Y_q(k^\star) \left(\Sigma^{-1} X_q^D - Y_q(k^\star)\Sigma^{-1}\alpha^D\right). \tag{3.20}$$

The objective function is extremal in $\hat{\alpha}^D$ if and only if its derivative is equal to zero at this point. Let Y^Q be an abbreviation for $Y^Q(ks)$. This condition takes the form of a

normal equation

$$\hat{\alpha}^D = \frac{\sum_{q=1}^{Q} Y_q X_q^D}{\sum_{q=1}^{Q} Y_q^2} = \frac{X^{D,Q}(Y^Q)^\intercal}{Y^Q(Y^Q)^\intercal}. \tag{3.21}$$

where the numerator is the inter-covariance matrix of $X^{D,Q}$ and Y^Q, and the denominator is the covariance matrix of Y^Q. □

Interestingly, the most likely value $\hat{\alpha}^D$ of α^D does not depend on the noise covariance matrix. As $N^{D,Q} = X^{D,Q} - \hat{\alpha}^D Y^Q$ has zero mean, the latter can be evaluated on its own as the well-known unbiased estimator of Σ:

$$\hat{\Sigma} = \tfrac{1}{Q-1}(X^{D,Q} - \hat{\alpha}^D Y^Q)(X^{D,Q} - \hat{\alpha}^D Y^Q)^\intercal. \tag{3.22}$$

By plugging *Eq. 3.21* into *Eq. 3.22*, one obtains

$$\begin{aligned}
\hat{\Sigma} &= \tfrac{1}{Q-1}\left(X^{D,Q} - X^{D,Q}\frac{(Y^Q)^\intercal Y^Q}{Y^Q(Y^Q)^\intercal}\right)\left(X^{D,Q} - X^{D,Q}\frac{(Y^Q)^\intercal Y^Q}{Y^Q(Y^Q)^\intercal}\right)^\intercal \\
&= \tfrac{1}{Q-1} X^{D,Q}\left(I^{Q,Q} - \frac{(Y^Q)^\intercal Y^Q}{Y^Q(Y^Q)^\intercal}\right)^2 (X^{D,Q})^\intercal \qquad (3.23) \\
&= \tfrac{1}{Q-1} X^{D,Q}\left(I^{Q,Q} - \frac{(Y^Q)^\intercal Y^Q}{Y^Q(Y^Q)^\intercal}\right) (X^{D,Q})^\intercal \qquad (3.24) \\
&= \tfrac{1}{Q-1}\left(X^{D,Q}(X^{D,Q})^\intercal - \frac{X^{D,Q}(Y^Q)^\intercal Y^Q (X^{D,Q})^\intercal}{Y^Q(Y^Q)^\intercal}\right).
\end{aligned}$$

In Eqn. (3.23), $I^{Q,Q}$ denotes the $Q \times Q$ identity matrix, and we use in *Eq. 3.24* the fact that the matrix $I^{Q,Q} - (Y^Q)^\intercal Y^Q / (Y^Q(Y^Q)^\intercal)$ is idempotent, i.e., equal to its square.

Remark 3.3. *We have the following remarkable identity:*

$$X^{D,Q}(X^{D,Q})^\intercal \quad = \quad \hat{\alpha}^D(\hat{\alpha}^D)^\intercal \; Y^Q(Y^Q)^\intercal \quad + \quad (Q-1)\hat{\Sigma}.$$

This equation is the non-asymptotic version of Proposition 3.5.

Comment 3.2. *This first characterization approach aims at computing the leakage model parameters. Thanks to the least-square estimation method we start by the computation of* $\widetilde{\alpha^D}$ *that is a estimation of* α^D *:*

$$\widetilde{\alpha^D} = XY^\intercal(YY^\intercal)^{-1}$$

then we compute $\widetilde{\Sigma}$ *the estimation of* Σ *regarding to* $\widetilde{\alpha^D}$*:*

$$\widetilde{\Sigma^D} = \mathbb{E}\Big[\Big(\widetilde{N} - \mathbb{E}\big[\widetilde{N}\big]\Big)\Big(\widetilde{N} - \mathbb{E}\big[\widetilde{N}\big]\Big)^\intercal\Big],$$

where $\widetilde{N} = X^{D,Q} - \widetilde{\alpha^D} Y^{Q\intercal}$ *is a matrix that store an estimation of the noise of each traces.*

3.6.2 SNR Computations of DPA Contest V2 [73] Traces of an AES Last Round

The values $\hat{\alpha}^D$ and $\hat{\Sigma}$ are represented in *Fig. 3.2*. We obtain:

- $\max_D \hat{\alpha}_d^2 / \hat{\Sigma}_{d,d} = 1.69 \cdot 10^{-3}$ (no dimensionality reduction)
- $\mathrm{SNR}_{\mathrm{PCA}} = \dfrac{((\hat{\alpha}^D)^\intercal \hat{\alpha}^D)^2}{(\hat{\alpha}^D)^\intercal \hat{\Sigma} \hat{\alpha}^D} = 1.36 \cdot 10^{-3}$ (PCA)
- $\mathrm{SNR}_{\mathrm{LDA}} = (\hat{\alpha}^D)^\intercal \hat{\Sigma} \hat{\alpha}^D = 12.78 \cdot 10^{-3}$ (LDA)

Therefore, the LDA has the largest SNR: it is about seven times larger than the maximum sample-wise SNR. The PCA has, in this example, an SNR smaller than the maximum univariable SNR (see *Lemma 3.1*).

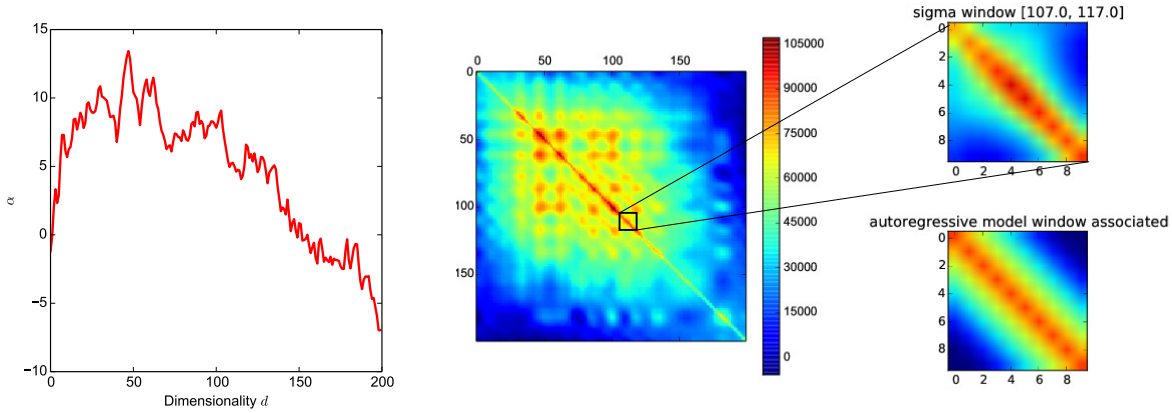


Figure 3.2: Estimated $\hat{\alpha}^D$ (*left*) and $\hat{\Sigma}^{D,D}$ (*right*), with $Q = 10,000$ traces, for locally autoregressive noise.

Interestingly, one can see in *Fig. 3.2* that the noise is locally autoregressive, for instance between samples 107 and 117.

3.7 Conclusions and Perspectives

Dimensionality reduction is common practice in SCA. This preprocessing technique has many virtues, such as an elegant multivariate description of the leakages, the concentration of information which reduces the required number of measurements to extract the key, and the increase of computational efficiency. Nonetheless, as any processing, dimensionality reduction can only reduce some information.

Using a mathematical approach, we have shown that dimensionality reduction is actually part of the optimal attack. This proves rigorously that dimensionality reduction can be achieved without loss in terms of attack success probability in extracting a secret key. As it turns out, the optimal dimensionality reduction consists in a linear projection of the trace samples.

We have also shown that the linear discriminant analysis asymptotically achieves the same projection, and hence becomes optimal as the number of traces increases. When the various samples are weakly correlated, we have found that PCA is nearly equivalent to the optimal dimensionality reduction and to LDA. Thus, in realistic contexts, state-of-the-art dimensionality reduction techniques are actually close to the optimal method.

Finally, we show how to estimate the model parameters (modulation vector α^D and noise covariance matrix Σ), and compute them on a real traces. An SNR gain factor of 7 can be obtained with respect to sample-wise SNR, which stresses the practical interest of dimensionality reduction.

As a perspective, we note that it should also be possible to obtain similar results when the noise is non-Gaussian (e.g., uniform). It is also desirable to compare dimensionality reduction based on linear projections to machine-learning techniques which are also multidimensional, such as SVM, random forests, K-means, ...

CHAPTER 4

Multivariate Leakages and Multiple Models

Contents

We have just tackled in the previous *Chap. 3* the problem of the optimal dimension reduction in the context of univariate model. In the present *Chap. 4* we generalize our solution to the multivariate models.

4.1 Contributions

Side-Channel Analysis (SCA) allow to extract secret keys from cryptographic devices. Template Attack (TA) [18] have been introduced as the strongest analysis method. They consist in two phases: (*i*) a profiling offline phase were the leakage model of the device under attack is characterized; (*ii*) an attack online phase in which the secret key is extracted using fresh measurements along with the precharacterized model. Such

attacks are known to use a maximum likelihood principle to ensure the highest possible success probability (see, eg., [42]).

In this chapter we study optimal attacks with the best possible success probability when extracting the secret key. The success probability in key recovery is chosen as a figure of merit for optimization. Such an objective is typical of "pure" SCA. Other approaches [35, 52, 79] relax the condition that the key found by the SCA be ranked first and complements it with a key enumeration stage. This yields a data vs. complexity trade-off that is not explored in this chapter. We leverage on such optimal distinguishers to answer the following question: "how to attack with the best probability of success when the leakage is multivariate and the model are multiple?" An initial empirical[1] work has already been carried out in [72] which confirmed that this type of approach can be very fruitful. Multi-target attacks [52, 80] have a somewhat different goal, namely the best aggregation of information about several subparts of a key, possibly leaking at different times with different models, in order to recover the full key efficiently. Here we consider only one multivariate leakage model and focus on recovering one subpart of the key. However, our derivation is capable of handling multivariate leakages and models and may still be combined with the multi-target approaches.

We derive closed-form expressions for the optimal distinguishers in all situations where the model is known (e.g., using profiling) or regressed online. In the case of a known univariate model, we recover the results in [13], However, our "fully matrix" formalism makes equations simpler and proofs shorter. Moreover, compared to [13] we extend the leakage model to the case where the traces are not necessarily centered, thereby allowing a more natural application on real traces. In the realistic "(on-line) stochastic attack" situation where the model is parametric, i.e. where the coefficients of the model are unknown, we express the optimal distinguisher by maximizing success over the whole set of possible coefficients. Finally, we provide fast computation algorithms for our novel distinguishers, which happen to be remarkably simple and efficient.

[1] The work in [72] does not detail the *modus operandi* result for the regression neither plugs it into the distinguisher, which is incidentally not chosen to be the optimal one.

4.2 Theoretical Results and Implementation

4.2.1 General Mathematical Expressions

In this section we derive the mathematical expression of the optimal distinguisher $\mathcal{D}$ in the general case of multivariate leakage ($D \geq 1$), and multiple models ($S \geq 1$). An illustration of our results is given in *Fig. 4.1* for the case when the leakage is completely known (or profiled as in the TA) and when the leakage is unknown and estimated online.

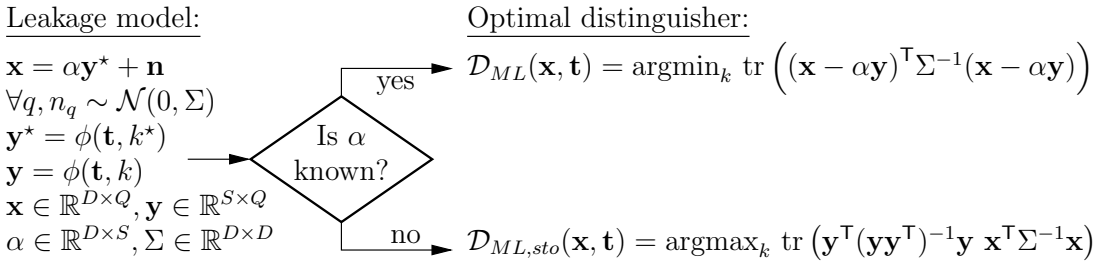


Figure 4.1: Mathematical expression for multivariate ($D \geq 1$) optimal attacks with a linear combination of models ($S \geq 1$)

Definition 4.1 (Optimal Distinguisher Knowing or Ignoring α)**.**

$$\mathcal{D}_{\text{ML}}(X, T) = \underset{k\in\mathbb{F}_2{}^n}{\text{argmax}}\, p(X|T) \qquad \textit{and}$$

$$\mathcal{D}_{\text{ML,sto}}(X, T) = \underset{k\in\mathbb{F}_2{}^n}{\text{argmax}} \max_{\alpha\in\mathbb{R}^{D\times S}} p(X|T, \alpha).$$

In both cases (Theorems 4.1 and 4.2 below) the result is a distinguisher which is computed using simple matrix operations. While $\mathcal{D}_{\text{ML}}$ resembles a TA with Gaussian templates [18], $\mathcal{D}_{\text{ML,sto}}$ is a novel expression that results from a non-trivial maximization over the matrix α and may be interpreted as a generalization of a multivariate Correlation Power Analysis (CPA) [11].

Theorem 4.1. *The optimal Maximum Likelihood (ML) distinguisher [42] for Gaussian noise writes*

$$\mathcal{D}_{\text{ML}}(X, T) = \underset{k}{\text{argmin}}\ \ tr\left((X - \alpha Y)^\mathsf{T}\Sigma^{-1}(X - \alpha Y)\right). \tag{4.1}$$

Proof. From [42] we have $\mathcal{D}_{\mathrm{ML}}(X,T) = \underset{k}{\operatorname{argmax}}\ p(X|Y)$ while from *Eq. 2.3* we see that $p(X|Y) = p_N(X - \alpha Y)$. From the i.i.d. assumption the noise density $p_N(n)$ is given by

$$\begin{aligned} p_N(n) &= \prod_{q=1}^{Q} \frac{1}{\sqrt{(2\pi)^D |\det \Sigma|}} \exp -\frac{1}{2} {n_q}^{\mathsf{T}} \Sigma^{-1} n_q \\ &= \frac{1}{(2\pi)^{DQ/2}} \frac{1}{(\det \Sigma)^{Q/2}} \exp -\frac{1}{2} \left(\sum_{q=1}^{Q} {n_q}^{\mathsf{T}} \Sigma^{-1} n_q \right) \\ &= \frac{1}{(2\pi)^{DQ/2} (\det \Sigma)^{Q/2}} \exp -\frac{1}{2} \mathrm{tr}\ (n^{\mathsf{T}} \Sigma^{-1} n). \end{aligned}$$

Thus $p_N(X - \alpha Y)$ is maximum when the expression $\mathrm{tr}\ (n^{\mathsf{T}} \Sigma^{-1} n)$ for $n = X - \alpha Y$ is minimum. □

In *Eq. 4.1* of *Theorem 4.1*, the trace

$$\mathrm{tr}\ ((X - \alpha Y)^{\mathsf{T}} \Sigma^{-1} (X - \alpha Y))$$

consists in:

- the sum of Q Mahalanobis [51] distances (see also Eq. (22) of [57]),
- the sum of D elements (which is useful when $D \ll Q$), as attested by rewriting

$$\mathrm{tr}\left(\underbrace{(X - \alpha Y)^{\mathsf{T}} \Sigma^{-1} (X - \alpha Y)}_{Q \times Q \text{ matrix}} \right) = \mathrm{tr}\left(\underbrace{\Sigma^{-1} (X - \alpha Y)(X - \alpha Y)^{\mathsf{T}}}_{D \times D \text{ matrix}} \right).$$

Theorem 4.2. *The optimal stochastic multivariate attack is given by*

$$\mathcal{D}_{\mathrm{ML,sto}}(X,T) = \underset{k \in {\mathbb{F}_2}^n}{\operatorname{argmax}}\ tr\ (Y^{\mathsf{T}} (Y Y^{\mathsf{T}})^{-1} Y \quad X^{\mathsf{T}} \Sigma^{-1} X) \tag{4.2}$$

for which the optimal value of α *is given by*

$$\alpha^{opt} = (X Y^{\mathsf{T}})(Y Y^{\mathsf{T}})^{-1}. \tag{4.3}$$

For the proof, we need some known results of linear algebra (*Lemma 4.1*) and linear regression (*Lemma 4.2*).

Lemma 4.1. *Let b an $S \times Q$ matrix, with $S < Q$. The $S \times S$ matrix bb^T is invertible if and only if b has full rank S, i.e., if and only if the S lines of b are independent.*

Proof. Let x be a $S \times 1$ column vector. We have that $x^\mathsf{T} bb^\mathsf{T} x = \|b^\mathsf{T} x\|^2 = 0$ implies $b^\mathsf{T} x = 0$ hence $x = 0$. Hence the matrix bb^T is positive definite. □

Lemma 4.2. *Let a, b and α be respectively $1 \times Q$, $S \times Q$ and $1 \times S$ with $S < Q$, where b has full rank S. Then $\|a - \alpha b\|_2$ reaches its minimum for $\alpha = ab^\mathsf{T}(bb^\mathsf{T})^{-1}$.*

Proof. Expanding the squared norm gives $\|a - \alpha b\|_2^2 = (a - \alpha b)(a - \alpha b)^\mathsf{T} = aa^\mathsf{T} - 2\alpha ba^\mathsf{T} + \alpha bb^\mathsf{T}\alpha^\mathsf{T}$.

Therefore, the gradient $\dfrac{\partial}{\partial \alpha}\|a - \alpha b\|_2^2 = -2ba^\mathsf{T} + 2bb^\mathsf{T}\alpha^\mathsf{T}$ vanishes if and only if $\alpha^\mathsf{T} = (bb^\mathsf{T})^{-1} ba^\mathsf{T}$, i.e., $\alpha = ab^\mathsf{T}(bb^\mathsf{T})^{-1}$ where we have used the fact that bb^T is invertible by *Lemma 4.1*. □

Proof of *Theorem 4.2*. Let $X' = \Sigma^{-1/2} X$ and $Y' = (YY^\mathsf{T})^{-1/2} Y$. The optimal distinguisher minimizes the following expression over $\alpha \in \mathbb{R}^{D \times S}$:

$$\begin{aligned} &\mathrm{tr}\,((X - \alpha Y)^\mathsf{T} \Sigma^{-1} (X - \alpha Y)) \\ &= \mathrm{tr}\,((X' - \alpha' Y)(X' - \alpha' Y)^\mathsf{T}) = \sum_{d=1}^{D} \|X'_d - \alpha'_d Y\|^2. \end{aligned}$$

By *Lemma 4.2* the minimization over α'_d yields $\alpha'_d = (X'_d Y^\mathsf{T})(YY^\mathsf{T})^{-1}$ for all $d = 1, \ldots, D$. This gives $\alpha' = (X'Y^\mathsf{T})(YY^\mathsf{T})^{-1}$ hence $\alpha = (XY^\mathsf{T})(YY^\mathsf{T})^{-1}$, which remarkably does *not* depend on Σ.

The minimized value of the distinguisher is thus

$$\begin{aligned}
&\min_{\alpha} \ \mathrm{tr}\ ((X-\alpha Y)^{\mathsf{T}}\Sigma^{-1}(X-\alpha Y)) \\
&= \mathrm{tr}\ ((X-\alpha^{\mathrm{opt}}Y)^{\mathsf{T}}\Sigma^{-1}(X-\alpha^{\mathrm{opt}}Y)) \\
&= \mathrm{tr}\ ((\mathsf{Id}-Y^{\mathsf{T}}(YY^{\mathsf{T}})^{-1})^2 X^{\mathsf{T}}\Sigma^{-1}X) \\
&= \mathrm{tr}\ (X^{\mathsf{T}}\Sigma^{-1}X) - \mathrm{tr}\ (Y^{\mathsf{T}}(YY^{\mathsf{T}})^{-1}\ X^{\mathsf{T}}\Sigma^{-1}X)
\end{aligned}$$

where Id denotes the $D \times D$ identity matrix and where $\mathrm{tr}\ (X^{\mathsf{T}}\Sigma^{-1}X)$ is a constant independent of k. This proves *Theorem 4.2*. □

The expression of $\mathcal{D}_{\mathrm{ML,sto}}(X,T)$ given in *Theorem 4.2* consists in the trace of a $Q \times Q$ matrix, which can be admittedly very large. It can be, however, rewritten in a way that is easier to compute when Q is much greater than S and D:

Corollary 4.1 (Alternative Expression of $\mathcal{D}_{\mathrm{ML,sto}}$)**.** *Letting $X' = \Sigma^{-1/2}\ X$, and $Y' = (YY^{\mathsf{T}})^{-1/2}\ Y$ as in the proof of Theorem 4.2, we have*

$$\mathcal{D}_{\mathrm{ML,sto}}(X,T) = \underset{k\in\mathbb{F}_2{}^n}{\mathrm{argmax}}\ \|X'Y'^{\mathsf{T}}\|_F. \tag{4.4}$$

Here the Frobenius norm is of a $D \times S$ matrix.

Proof. Let us write $(YY^{\mathsf{T}})^{-1} = (YY^{\mathsf{T}})^{-1/2}(YY^{\mathsf{T}})^{-1/2}$ in *Eq. 4.2*. By the properties of the trace,

$$\begin{aligned}
&\mathrm{tr}\ (Y^{\mathsf{T}}(YY^{\mathsf{T}})^{-1}Y\ \ X^{\mathsf{T}}\Sigma^{-1}X) \\
&= \mathrm{tr}\left(\underbrace{(YY^{\mathsf{T}})^{-\frac{1}{2}}Y(\Sigma^{-\frac{1}{2}}X)^{\mathsf{T}}}_{S\times D}\underbrace{\left((YY^{\mathsf{T}})^{-\frac{1}{2}}Y(\Sigma^{-\frac{1}{2}}X)^{\mathsf{T}}\right)^{\mathsf{T}}}_{D\times S}\right) \\
&= \mathrm{tr}\left((Y'X'^{\mathsf{T}})(Y'X'^{\mathsf{T}})^{\mathsf{T}}\right) = \|X'Y'^{\mathsf{T}}\|_F{}^2.
\end{aligned}$$

□

Remark 4.1. *Notice that in Corollary 4.1, Y' is a vector of empirical covariance equal to the identity matrix. Indeed, $Y'Y'^{\mathsf{T}} = (YY^{\mathsf{T}})^{-1/2}YY^{\mathsf{T}}(YY^{\mathsf{T}})^{-1/2} = \mathsf{Id}$.*

4.2.2 Mathematical Expressions for $S = 2$

In order to provide interpretations for the optimal distinguisher expressions, we detail how an optimal attack unfolds when the leakage consists in a sum of a modulated scalar model and an offset ($S = 2$). The cases for profiled attacks (denoted $\mathcal{D}_{\text{ML}}^{S=2}$) and non-profiled attacks (denoted $\mathcal{D}_{\text{ML,sto}}^{S=2}$) are presented in *Fig. 4.2*.

Interestingly, when $S = 2$, the TA can decompose in two steps (affine projection followed by a Euclidean distance to the model). Remarkably, the projection vector is the same for all key guesses. This extends similar results [13] where only the linear relationship between leakage and model is explored. As for the online attack, $\mathcal{D}_{\text{ML,sto}}^{S=2}$ consists in a sum of square of CPA on transformed data, aiming at orthogonalizing the noise.

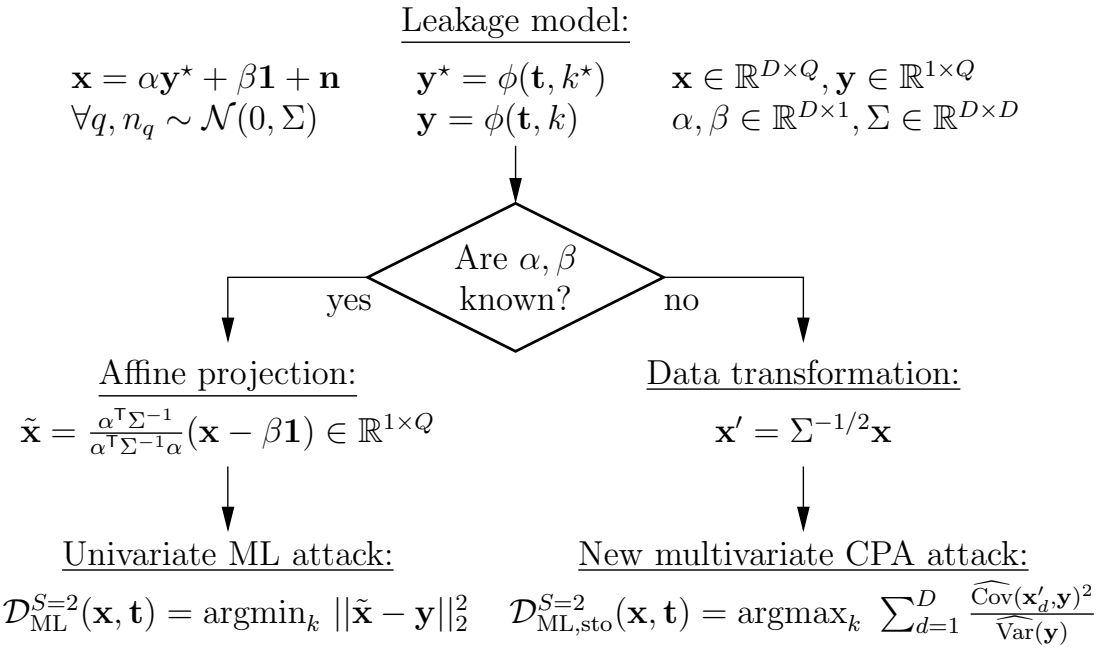


Figure 4.2: Modus operandi for multivariate ($D \geq 1$) optimal attacks with one model Y associated to envelope $\alpha \in \mathbb{R}^{D\times 1}$ and a constant offset $\beta \in \mathbb{R}^{D\times 1}$ ($S = 2$)

4.2.3 Efficient Implementation

Both $\mathcal{D}_{\mathrm{ML}}$ and $\mathcal{D}_{\mathrm{ML,sto}}$ can be optimized using the idea presented in [49]. This article applies a precomputation step in the case the number of traces is larger than the number of possible plaintexts ($Q > \#T = 2^n$). In this case, all summations $\sum_q$ can be advantageously replaced by $\sum_T \sum_{T_q=t}$. In most cases, the sum $\sum_{T_q=t}$ can be achieved on the fly, and does not involve an hypothesis on the key. Therefore, a speed gain of 2^n (the cardinality of the key space) is expected.

Such optimization strategy can be applied to $\mathcal{D}_{\mathrm{ML}}$. Indeed, let us define $X' = \Sigma^{-1/2}X$ and $\alpha' = \Sigma^{-1/2}\alpha$. Then,

$$\mathcal{D}_{\mathrm{ML}}(X,T) = \underset{k}{\operatorname{argmin}} \sum_{d=1}^{D} \|X'_d - \alpha'_d Y\|_2^2 \qquad \text{(see } \textit{Corollary 4.1}\text{)}$$

$$= \underset{k}{\operatorname{argmin}} \sum_{d=1}^{D} \sum_{T\in\mathbb{F}_2{}^n} \left(\sum_{q/T_q=t} {X'_{d,q}}^2 - 2 \sum_{q/T_q=t} X'_{d,q}\alpha'_d Y(t,k) + \Big(\sum_{q/T_q=t} 1\Big)(\alpha'_d Y(t,k))^2 \right)$$

$$= \underset{k}{\operatorname{argmin}} \sum_{d=1}^{D} \sum_{T\in\mathbb{F}_2{}^n} -2 \underbrace{\Big(\sum_{q/T_q=t} X'_{d,q}\Big)}_{\text{denoted as } X'_{d,t}} \alpha'_d Y(t,k) + \underbrace{\Big(\sum_{q/T_q=t} 1\Big)}_{\text{denoted as } n_T} (\alpha'_d Y(t,k))^2 \tag{4.5}$$

$$= \underset{k}{\operatorname{argmax}} \ \operatorname{tr}\left(X'\big(\alpha' Y(k)\big)^{\mathsf{T}}\right) - \frac{1}{2}\sum_{T\in\mathbb{F}_2{}^n} n_T \ \big\|\alpha' Y(T,k)\big\|_2^2 . \tag{4.6}$$

Notice that at line *Eq. 4.5*, the term $\sum_{q/T_q=t} {X'_{d,q}}^2$ which does not depend on the key, is simplified. The fast version of this computation is given in *Alg. 4.1*.

The same optimization applies to $\mathcal{D}_{\mathrm{ML,sto}}$. Indeed, in expression (4.4) of $\mathcal{D}_{\mathrm{ML,sto}}(X,T) = \underset{k}{\operatorname{argmax}} \ \|X'Y'^{\mathsf{T}}\|_F^2$, one can write

$$\|X'Y'^{\mathsf{T}}\|_F^2 = \sum_{s,d}\Big(\sum_{q=1}^{Q} X'_{d,q}Y'_{s,q}\Big)^2$$

$$= \sum_{s,d}\Big(\sum_{t\in\mathbb{F}_2{}^n} \underbrace{\Big(\sum_{q/T_q=t} X'_{d,t}\Big)}_{\text{denoted as } X'_{d,t}} \underbrace{\big(Y'_s(t,k)\big)}_{\text{denoted as } Y'_{s,t}} \Big)^2 . \tag{4.7}$$

Algorithm 4.1: Fast computation algorithm for $\mathcal{D}_{\mathrm{ML}}$

input : X, T

output: $\mathcal{D}_{\mathrm{ML}}(X, T)$

// Initialize to zero a matrix X' of size $D \times 2^n$

// Initialize to zero a vector n_t of length 2^n

1 **for** $q \in \{1, \ldots, Q\}$ **do** // On-the-fly accumulation

2 $\quad X'_{T_q} \leftarrow X'_{T_q} + \Sigma^{-1/2} X_q$

3 $\quad n_{T_q} \leftarrow n_{T_q} + 1$

// Single evaluation, as in (4.6)

4 **return** $\underset{k \in \mathcal{K}}{\operatorname{argmax}}\ \operatorname{tr}\left(X'\left(\alpha' Y(k)\right)^{\mathsf{T}}\right) - \frac{1}{2}\sum_t n_t \left\|\alpha' Y(t,k)\right\|_2^2$

This means that X' can be obtained by simple accumulation, exactly as in *line 2* of *Alg. 4.1*. The term $Y'_s(t,k)$ requires the computation of YY^{T}. In the case $Q \gg 1$, it can be assumed that the texts T are uniformly distributed. Hence, when $Q \to +\infty$, by the law of large numbers,

$$\frac{1}{Q} YY^{\mathsf{T}} = \frac{1}{Q}\sum_{q=1}^{Q} Y_q {Y_q}^{\mathsf{T}} = \sum_{t \in {\mathbb{F}_2}^n} \frac{\sum_{q/T_q = t} 1}{Q} Y(t,k) Y(t,k)^{\mathsf{T}}$$
$$\xrightarrow[Q \to +\infty]{} \frac{1}{2^n} \sum_{t \in {\mathbb{F}_2}^n} Y(t,k) Y(t,k)^{\mathsf{T}}.$$

Therefore, in (4.7), $Y'_s(t)$ can also be precomputed. To the best of our knowledge, this optimization has never been discussed previously. The resulting distinguishing procedure is given in Alg. 4.2. At line 3, the argument of the Frobenius norm can be computed by a fast matrix multiplication. Also, we notice that the matrix inversion at line 0 is actually a precomputation which involves only the model. Besides, if the EIS (*Equal Images under all Sub-keys*) assumption holds [67, Def. 2], e.g., $Y(t,k)$ only depends on $t \oplus k$, then $\sum_t Y(t,k)Y(t,k)^{\mathsf{T}}$ does not depend on k, hence only one single matrix inversion to compute. Eventually, the computational complexity of the optimal stochastic attack simply consists in traces accumulation per class, and as many matrix products and Frobenius norms as keys to be guessed.

Algorithm 4.2: Fast computation algorithm for $\mathcal{D}_{\mathrm{ML,sto}}$ when T is balanced

input : X, T

output: $\mathcal{D}_{\mathrm{ML,sto}}(X, T)$

// Precompute $\#\mathcal{K} = 2^n$ matrices $Y'(T,k)$ of size $S \times 2^n$, s.t. $Y'(T,k) = (\frac{1}{2^n}\sum_t Y(t,k)Y(t,k)^\mathsf{T})^{-1/2}Y(T,k)$.

// Initialize to zero a matrix $X'_{d,t}$ of size $D \times 2^n$

for $q \in \{1, \ldots, Q\}$ **do**

  $X'_{t_q} \leftarrow X'_{t_q} + \Sigma^{-1/2}X_q$ // In-place accumulation of a column in matrix X'

return $\underset{k\in\mathcal{K}}{\mathrm{argmax}}\ \|X'Y'(T,k)^\mathsf{T}\|_F$ // As in (4.7)

4.3 Practical Results

4.3.1 Characterization of Σ

In this chapter, we assume that the attacker knows the noise covariance matrix. We give a straightforward procedure for the estimation.

1. collect Q traces (i.e., a matrix $X \in \mathbb{R}^{D\times Q}$) where the plaintext is fixed to a given value,

2. estimate Σ as $\hat{\Sigma} = \frac{1}{Q-1}\big(X - \frac{1}{Q}X\mathbf{1}^\mathsf{T}\mathbf{1}\big)\big(X - \frac{1}{Q}X\mathbf{1}^\mathsf{T}\mathbf{1}\big)^\mathsf{T}$, where $\mathbf{1} = (1, \ldots, 1) \in \mathbb{R}^{1\times Q}$. This estimator is *sample covariance matrix*, which is unbiased.

Remark 4.2. *Notice that Σ cannot be obtained by a direct profiling on the same traces to be used for the attack. Indeed, in those traces, the plaintext is varying, hence the attacker would use for $\hat{\Sigma}$ the covariance matrix of $X - \alpha^{opt}Y$, where α^{opt} is equal to $\alpha^{opt} = (XY^\mathsf{T})(YY^\mathsf{T})^{-1}$ (recall Eq. 4.3). Hence, $\hat{\Sigma} = \frac{1}{Q-1}(X - \alpha^{opt}Y)(X - \alpha^{opt}Y)^\mathsf{T}$.*

But the distinguisher $\mathcal{D}_{\text{ML,sto}}$ *is*

$$\begin{aligned}
&\mathcal{D}_{\text{ML,sto}}(X,T)\\
&= \underset{k\in\mathbb{F}_2{}^n}{\text{argmin}} \min_{\alpha\in\mathbb{R}^{D\times S}} tr\left((X-\alpha Y)^{\mathsf{T}}\hat{\Sigma}^{-1}(X-\alpha Y)\right)\\
&= \underset{k\in\mathbb{F}_2{}^n}{\text{argmin}} \min_{\alpha\in\mathbb{R}^{D\times S}} tr\left(\hat{\Sigma}^{-1}(X-\alpha Y)(X-\alpha Y)^{\mathsf{T}}\right)\\
&= \underset{k\in\mathbb{F}_2{}^n}{\text{argmin}}\ tr\left(\hat{\Sigma}^{-1}(X-\alpha^{opt}Y)(X-\alpha^{opt}Y)^{\mathsf{T}}\right) \qquad (4.8)\\
&= \underset{k\in\mathbb{F}_2{}^n}{\text{argmin}}\ tr\left((Q-1)\hat{\Sigma}^{-1}\hat{\Sigma}\right) = \underset{k\in\mathbb{F}_2{}^n}{\text{argmin}}\ D(Q-1). \qquad (4.9)
\end{aligned}$$

Indeed, at line 4.8, we demonstrated in the proof of Theorem 4.2 in that the minimal value in Eq. 4.3 of α *is independent on* Σ*. Eventually, it can be seen at line 4.9 that the distinguisher with* $\hat{\Sigma}$ *instead of* Σ *does not depend on the key. Indeed,* $\underset{k}{\text{argmin}}\ cst = \mathbb{F}_2{}^n$*, meaning that all keys are equiprobable. Intuitively, when both the noise and the model parameters are regressed at the same time, any key manages to achieve the same match between parametric model and Side-Channel (SC) observations.*

4.3.2 Attacks on Synthetic (i.e., Simulated) Traces

In this subsection we present simulations when α is known exactly or regressed online. We consider an attack of PRESENT, where the S-box is $n = 4 \to n = 4$. For the sake of the simulations, we choose two kinds of α:

- "identical": all the $n = 4$ bits leak the same waveform, like in the Hamming Weight (HW) model,
- "proportional": the waveform has weight 1 for S-box bit 0, and is multiplied by 2 (resp. 3 and 4) for S-box bit 1 (resp. 2 and 3).

The waveform for each bit is that represented in *Fig. 2.5* (upper left graph). Specifically, for all $1 \leq d \leq D$ and $1 \leq s \leq S$, the envelope consists in damped oscillations:

$$\alpha_{d,s} = e^{-\frac{2d}{D}}\cos\left(2\pi\frac{d}{D}\right) \quad \text{for the "identical" case,} \qquad (4.10)$$

$$\alpha_{d,s} = s\cdot e^{-\frac{2d}{D}}\cos\left(2\pi\frac{d}{D}\right) \quad \text{for the "proportional" case.} \qquad (4.11)$$

The noise is chosen normal, using two distributions:

- "isotropic": the covariance matrix is σ^2 times the $D \times D$ identity,
- "autoregressive": the covariance matrix element at position (d, d'), for $1 \leq d, d' \leq D$, is $\sigma^2 \rho^{|d-d'|}$. This noise is not independent from sample to its neighbours, but the correlation ρ decreases exponentially as samples get further apart.

Proposition 4.1. *The success probability of $\mathcal{D}_{\mathrm{ML}}$ is greater than that of $\mathcal{D}_{\mathrm{ML,sto}}$.*

Proof. Indeed, according to [42], $\mathcal{D}_{\mathrm{ML}}$ maximizes the success probability. Thus, the distinguisher $\mathcal{D}_{\mathrm{ML,sto}}$ has a smaller success probability. The success probability is the same if the minimization over α in the proof of *Theorem 4.2* yields the exact matrix α used in the model provided in *Eq. 2.3*. □

Simulations allow to estimate the loss in terms of efficiency of not knowing the model (*Proposition 4.1*), by comparing distinguishers $\mathcal{D}_{\mathrm{ML}}$ (*Eq. 4.1*) and $\mathcal{D}_{\mathrm{ML,sto}}$ (*Eq. 4.2*). The success rate of the optimal distinguisher $\mathcal{D}_{\mathrm{ML}}$ is drawn in order to materialize the limit between feasible (below) and unfeasible (above) attacks.

Results for low noise ($\sigma = 1$) are represented in *Fig. 4.3*. We can see that the HW model is clearly harder to attack, because the leakage of one bit cannot be distinguished from that of the other bits. Besides, we notice that the stochastic attack is performing much worse than the optimal attack: about 10 times more traces are required for an equivalent success probability in key extraction. Results for high noise ($\sigma = 4$) are represented in *Fig. 4.4*. Again, the "proportional" model is easier to attack than the "identical" model (for each bit). Now, we also see that the gap between the optimal ML attack and the stochastic attack narrows: only about 5 times more traces are needed for the stochastic attack to perform as well as the optimal attack in terms of success probability. Besides, we notice that the autoregressive noise is favorable to the attacker. It is therefore important in practice for the attacker to characterize precisely the noise distribution (recall the methodology presented in *Subsec. 4.3.1*).

Clearly, these conclusions are in line with the "template versus stochastic" (offline) study carried out in [34]: for high noise, the (online) learning of the model requires more traces, hence is more accurate. Therefore, the performance of $\mathcal{D}_{\mathrm{ML,sto}}$ gets closer to that of $\mathcal{D}_{\mathrm{ML}}$ than for high noise.

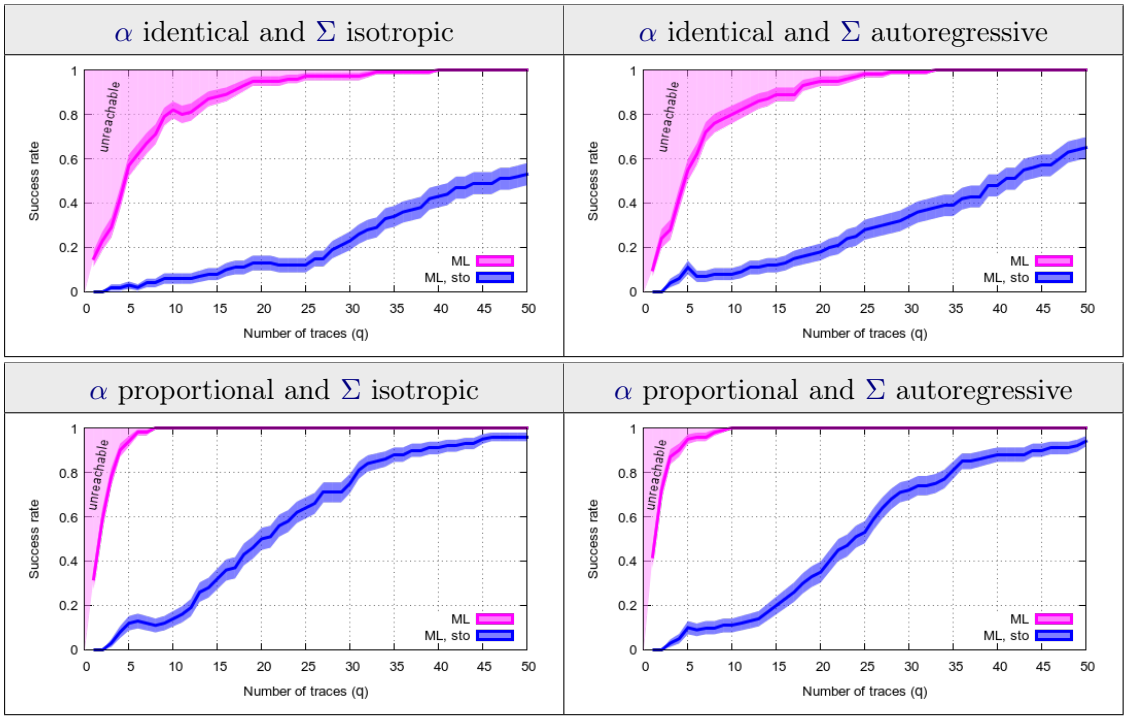


Figure 4.3: Simulations for $D = 3$, $S = 5$, $n = 4$, $\sigma = 1$ (autoregressive noise with $\rho = 0.5$).

Simulations When α is Estimated Offline.

Furthermore, we used simulated traces to evaluate the influence of the number of traces in the learning set for α (when it is profiled offline) over the success rate of our attacks.

The result of this process is illustrated in *Fig. 4.5*, for a similar case study (Advanced Encryption Standard (AES) S-box is used instead of PRESENT). This study has been done with proportional and identical α, and isotropic and autoregressive noise. In the case of $S = 2$ the success rate is always higher than in the case of $S = 9$. In addition, for each size of learning set, the mean of the Mean Square Error (MSE) has been computed. The MSE for $S = 2$ converges more quickly to the value of the noise variance than for $S = 9$, which accounts for the fact that when S increases, more traces are required for the profiling of the model.

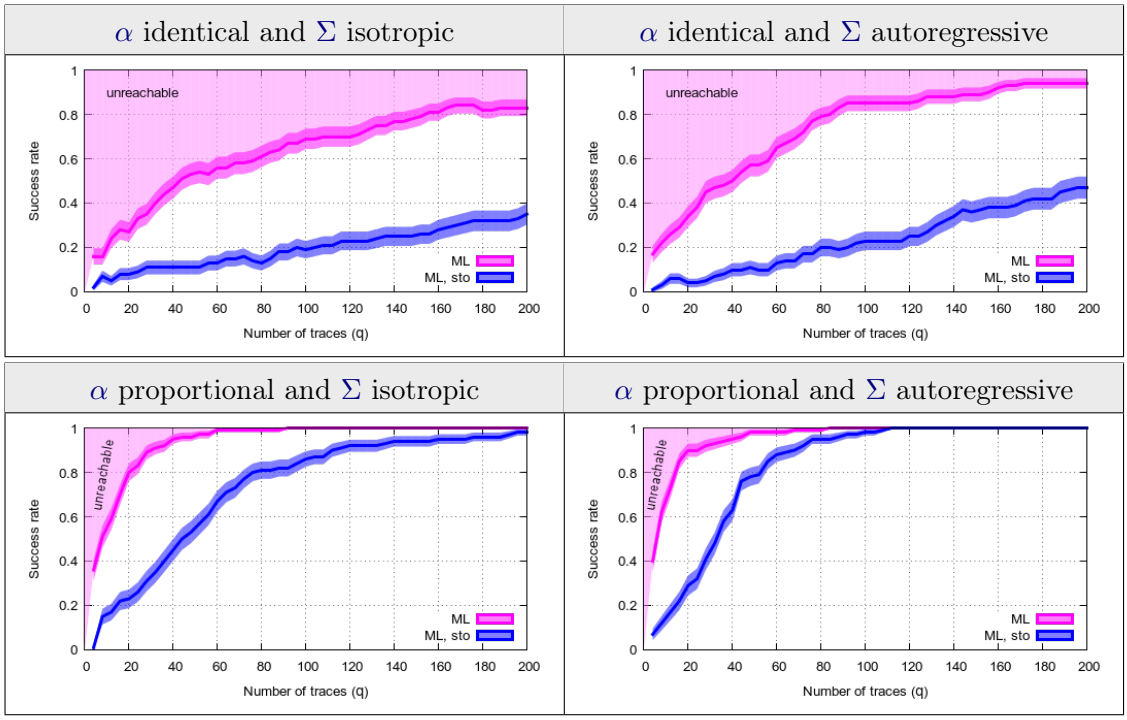


Figure 4.4: Simulations for $D = 3$, $S = 5$, $n = 4$, $\sigma = 4$ (autoregressive noise with $\rho = 0.5$).

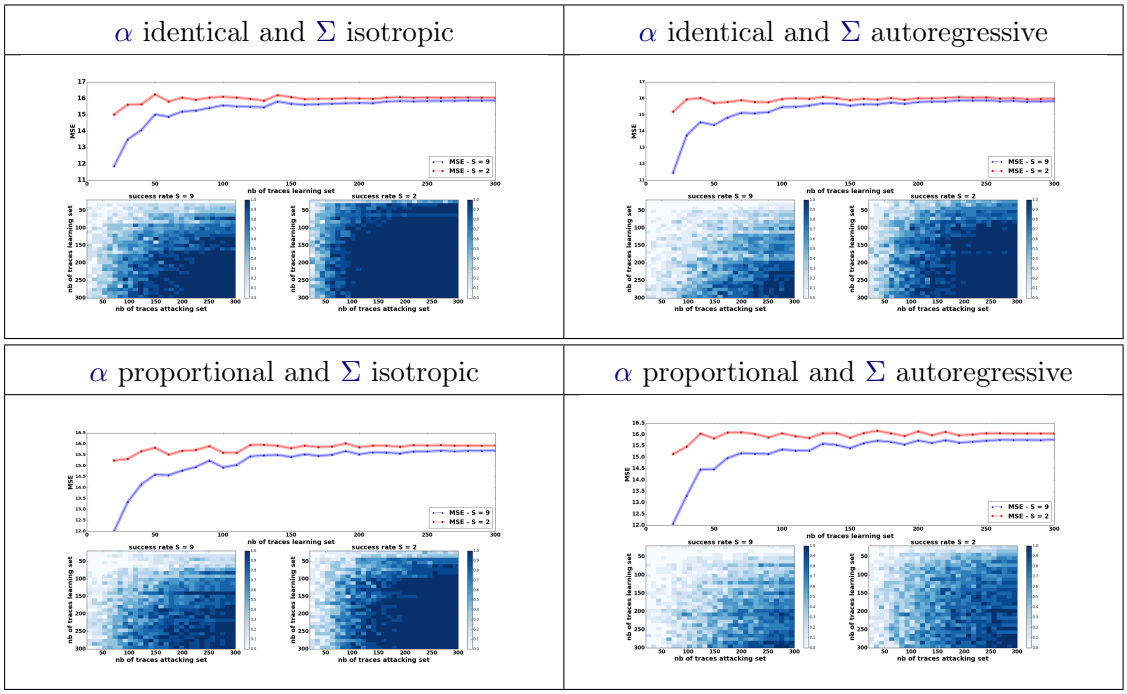


Figure 4.5: Influence of the number of traces in the learning set over the success rate and the corresponding MSE, for $D = 3$, $S = 5$, $n = 4$, $\sigma = 16$, and an autoregressive noise with $\rho = 0.5$.

4.3.3 Attacks on Real-World Traces

We now compare CPA with $\mathcal{D}_{\text{ML}}$ and $\mathcal{D}_{\text{ML,sto}}$ on measurements provided by the DPA contest V4 [74] . These traces have been acquired from an 8-bit processor, hence have a Signal-to-Noise Ratio (SNR) greater than one, reaching 7 at some points in time. The interval for our case-study is $[170, 250]$ from *Fig. 2.6*, hence $D = 80$. Regarding ML, two learning strategies have been implemented:

1. the model is learned from a disjoint set of 5k traces, which is the operational scenario for a profiled attack;
2. the model is learned from the traces being attacked (denoted **self** in *Fig. 4.6*). This case does not represent a realistic attack, but is interesting in that it highlights the best possible attacker.

The attack success rates are plotted in *Fig. 4.6*. One can see that both variants of $\mathcal{D}_{\text{ML}}$

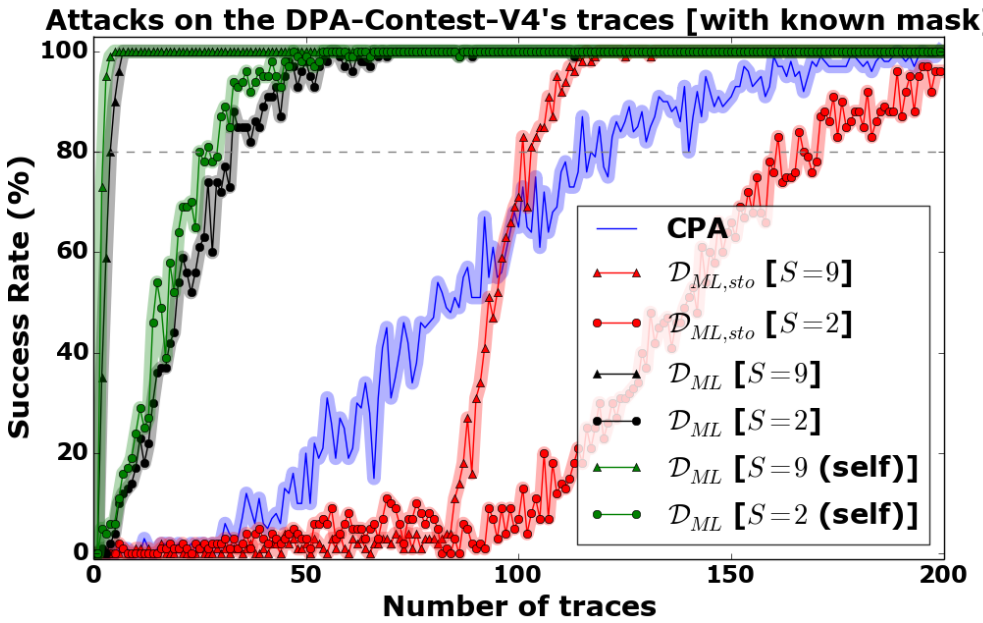


Figure 4.6: Success rate of CPA, $\mathcal{D}_{\text{ML,sto}}$, and $\mathcal{D}_{\text{ML}}$ for $S \in \{9, 2\}$ (with two distinct learning methods)

and $\mathcal{D}_{\text{ML,sto}}$ achieve better with $S = 9$ than with $S = 2$. This is consistent with the analysis carried out in *Subsec. 2.2.2*. Actually, the CPA has a very poor performance because the model is actually very far from the HW: as can be seen in *Fig. 2.6*, some parameters α_i (e.g., for $i = 2$ and 6) are positive in region $[180, 200]$ whereas others α_j (e.g., for $j = 1$, 3, 4 and 5) are negative. The compensating signs account why the HW model is inappropriate. The ML with model precharacterization on the traces under attack show that very strong attacks are possible (using a few traces only). Interestingly, when the model used by ML is characterized on 5k traces distinct from the traces being attacked, the performance is almost similar. Eventually, the online stochastic attack derived in this chapter ($\mathcal{D}_{\text{ML,sto}}$) performs better than CPA (the distinguisher being the maximum value of the Pearson correlation over the $D = 80$ samples).

4.4 Conclusions and Perspectives

Distinguishing a key from both multivariate leakage samples and multiple models can be done in one step as shown in this chapter. A compact expression of the distin-

guisher is provided, using matrix operations. The strategy is applied to real-world traces in profiled and non-profiled scenarios. The resulting attack is more efficient than the traditional approach "dimensionality reduction then stochastic (linear regression) attack)". The new multivariate distinguisher outperforms the other state-of-the-art attacks. The presented methodology allows for leakage agnostic attacks on vectorial leakage measurements and complex models. In addition, the matrix-based expression of the distinguisher benefits from matrix-oriented software that implements computational optimizations for large dimensions.

A companion future work would consist in determining the optimal model dimensionality and basis from any acquisition campaign. Another perspective is to adapt the methodology to masked implementations, as already done for univariate leakage in [24], yet for this case the distinguishers will certainly not exhibit simple closed-form expressions. However, we believe that the approach could be fruitful in practice backed with suitable optimization software.

CHAPTER 5

Binary Data Analysis for Source Code Leakage Assessment

Contents

We have just tackled in the previous *Chap. 4* the problem of optimal distinguisher in the context of multivariate leakage and models. In the present *Chap. 5* we propose an analysis of multivariate leakage and model on 3-dimensional "software traces" provided by debugger tool.

5.1 Introduction

5.1.1 Previous Work

Measuring ElectroMagnetic (EM) or power traces from embedded devices to identify potential leakage of information is a time consuming and challenging process. First,

it requires equipment (oscilloscope, probes, signal amplifiers...) that demand prior knowledge to the technician. Secondly, the code to evaluate needs to be embedded on the final product or on an evaluation board, it means that the development has to be finished at the evaluation time. This justifies the interest of the security evaluation community for simulation, both in the industrial and academic world. Aiming to speed-up, facilitate the evaluation, and allowing an evaluation during the development, several works have proposed Side-Channel (SC) trace simulators. The principle is to simulate the leakage that might happen during the code execution. Numerous simulators are available to simulate SC (power or EM) leakage. All present a similar general construction flow illustrated in *Fig. 5.1*. First, they take a description of the implementation to evaluate as input. For example, the inputs of SILK [78] are tagged C++ source codes. More often, inputs are architecture dependant compiled binaries: Elmo [53] uses binaries for the ARM Cortex-M0 and OSCAR [76] uses binaries for the 8-bit Atmel AVR microcontroller. The Dynamic Binary Analyzer (DBA) framework [3] supports more architectures: ARM, x86, MIPS, SPARC and SH4. In the simulation step, the SC-simulators execute the code to record data. For example Elmo [53] uses the emulator Thumbulator[1], while in [9] the authors use the Valgrind debugger. The choice of the data provider is influenced by the fact that the simulator is specific to an architecture. Another important choice of the simulation step is the selection of the data to record. The authors of [9] proposed to record the memory accesses. The DBA framework [3] records the stack, the heap, the CPU-registers and the executed instructions. Elmo [53] is focused on the values of the operands, the bit-flips of the operations and the operations. The last step is the trace generation, or how to transform the recorded data to obtained traces as similar as possible to real and effective physical leakage. Elmo [53] provides one of the most realistic and complex model computed with linear regressions (also used in [54]) and F-test on real leakage traces recorded on a ARM Cortex-M0. Otherwise, the most commonly used models are the Hamming Weight (HW) and the Hamming Distance (HD), which are simplifications of effective physical leakage as introduced in the *Subsec. 2.2.3*. Those models have been preferred by the authors of [3, 76].

[1] https://github.com/dwelch67/thumbulator

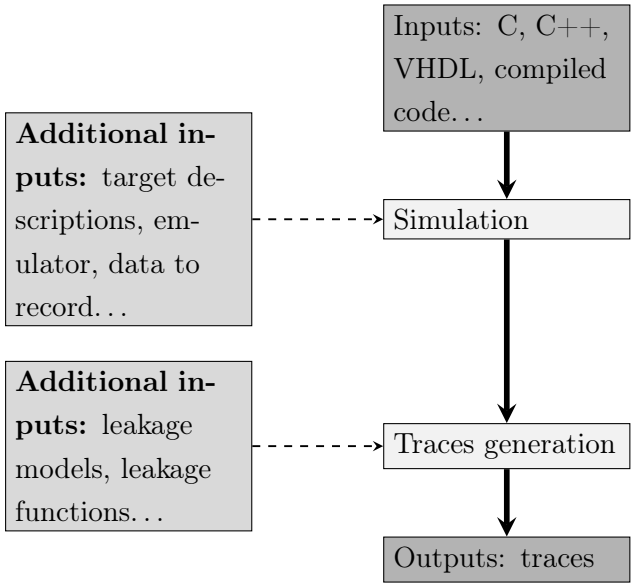


Figure 5.1: General flow commonly used by SC simulators.

5.1.2 Contributions

In this context, we propose a new methodology of data collection and analysis to identify potential leakages from any software implementation. Our solution takes inspiration from the work proposed by Bos *et al.* in [9] . An improvement of the analysis step has been proposed in [2], realizing on three intermediate computations instead of one initially. We investigate the proposed improvement in the *Subsec. 5.3.2*. As a first difference, we propose a new methodology of data collection. We record all bit-modifications that happen during a code execution (registers, memory content, flags, Program Counter (PC)), while in [9], authors record only the read, written and executed addresses. Furthermore, we do not apply any model to the recorded data in order to generate traces. All the collected data is exhaustively analyzed to avoid information loss. To do so, we introduce a binary analysis to extract the leaking points from the data. Then, we leverage on PC to keep track of the execution context to map the identified leakage to the source code. Indeed, mapping leakages to source code is nowadays an important need for continuous improvement of products; this is particularly true for software implementations, where traces are long and code is complex. To succeed, we need to overcome two main difficulties: misalignment and multiplicity of leaking resources. We show how to characterize and then exploit the PC to realign all the data using a simple accumulative algorithm, and we introduce a methodology of data

selection that significantly reduce the size of the data to analyze. Finally, we show how our solution can identify leakage in the source code applying our methodology to White Box Cryptography (WBC) implementation.

5.2 Solution Presentation

5.2.1 Notations and Recording Step

Aiming to record all the data manipulated by a given binary, the debugger GNU Debugger (GDB) is used, but alternative software could be used to collect data. The analysis that we propose is independent of the data provider. To identify all potential leakages, the recording process is as exhaustive as possible. Each time the PC changes, all the internal data are saved (PC, registers, flags...). For example, in the case of an x86 architecture in 64-bit execution (properties of the system used for all the results given in the current paper), the internal data are:

- the sixteen 64-bit registers: `rax, rbx, rcx, rdx, rsi, rdi, rbp, rsp, r8, r9, r10, r11, r12, r13, r14, r15`,
- the six 16-bit registers: `cs, ss, ds, es, fs, gs`,
- the 64-bit `eflags` (with the bit 1, 5, 15, 22-63 reserved).
- the PC (Program Counter, also named `rip` for register instruction pointer)

As in the previous chapters, the matrix notations are used. The recorded data of an execution is stored in a matrix noted $X^{D,R} \in (\mathbb{F}_2)^{D,R}$, with D the number of times the PC changes and R the number of bit needed to store all the internal data (except the PC that is stored independently, and used in resynchronization process deepened in *Subsec. 5.2.2*). For a given dataset $X^{D,R}$, the associated list of successive PC-values is stored in a matrix $\mathrm{Pc}^D \in (\mathbb{Z})^D$. An illustration of a recorded trace is provided in *Fig. 5.2*, the black color corresponds to one and the white to zero, the x-axis describes the internal data and the y-axis the index of the PC. The illustrated trace follows from the execution of the WBC algorithm freely provided at the Capture The Flag (CTF) of CHES-2016[1]. The illustration of the recorded trace provided in *Fig. 5.2* shows that only a little part of the registers seems to be used during the execution. This observation inspired us the

[1] http://ctf.newae.com/

data reduction algorithm described in *Subsec. 5.2.3*. A set of Q executions is noted as

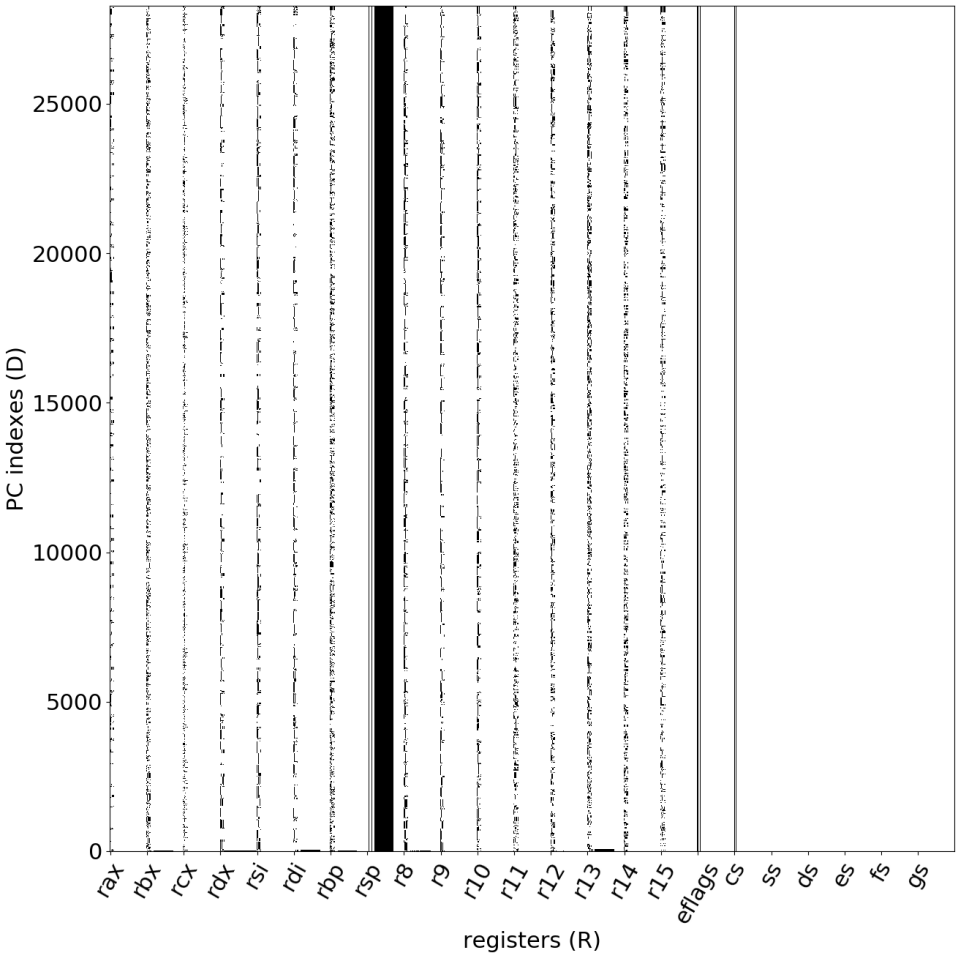


Figure 5.2: Software trace of a WBC algorithm using GDB, black points correspond to one values, and white to zero values.

$\{X_q^{D_q,R}, \mathrm{Pc}_q^{D_q}\}_{q<Q}$, an element of $X^{D,R}$ is noted $X_{d,r}$ and X_d^R (resp. X_r^D) refers to a row matrix (resp. a column matrix). In the context of the Side-Channel Analysis (SCA), the traces are commonly compared with distributions of intermediate values manipulated by the target and dependent of a secret variable (generally a cryptographic key). We store those distributions in the matrix $Y^{S,Q,K,B}$, with S the dimension of the leakage model, B the number of bytes of the secret and K the number of possible values for

each secret bytes (256 if no values are forbidden). For example, if the target algorithm is the AES-128 , and if the focused sensitive intermediate value is the output of the S-box at the first round, the distribution for a *bit-level* model is expressed as in the following *Formula 5.1*:

$$Y^{S,Q,K,B} = \left\{ (\text{S-box}(T_{q,b} \oplus k) \ \& \ 2^s) >> s \right\}_{\substack{s<8, k<256 \\ q<Q, b<16}} \tag{5.1}$$

The choice of the *bit-level* model is motivated by the bit representation of the recorded data $X^{D,R} \in (\mathbb{F}_2)^{D,R}$. To lighten the notations, the dimension B is not always precised.

5.2.2 Realignment Algorithm

The first problematic met in the proposed study is the misalignment of the data. Indeed, misalignment could be due to the randomization of the execution, or more generally, by the presence of conditional branches. A vertical alignment is a prerequisite point to realize vertical analysis. Most of the vertical analysis techniques, as Correlation Power Analysis (CPA) [11] or Linear Regression Analysis (LRA) [29] need the data $X_d^{R,Q}$ manipulated at the sample $d < D$ to come from the same operation. The resynchronization is a well known and a well studied problem in the SCA domain [28, 36, 75, 77]. All the proposed algorithms of resynchronization are based on the leaking values distribution in the temporal or in the frequency domain. In our case we have access to additional information thanks to the PC values. In fact, the PC values can be viewed as an identifier. For example $\forall q < Q, \forall d_q < D_q$, $\text{Pc}_{d_q,q}$ is an identifier for the data $X^R_{d_q,q}$. Furthermore, if for $d_0 < D_0, d_1 < D_1$, $\text{Pc}_{d_0,0} = \text{Pc}_{d_1,1}$ it means that the two datasets $X^R_{d_0,0}$ and $X^R_{d_1,1}$ are the result of the same operation in the code (at the assembly level). However the presence of a loop in the source code could imply repetitions of PC value. Hence evince, if the two datasets $X^R_{d_0,0}, X^R_{d_1,1}$ result form the same operation they may come from distinct iterations. Moreover, conditional branching in the code produce misalignment. The goal of the proposed realignment algorithm is to transform the raw dataset $\{X_q^{D_q,R}, \text{Pc}_q^{D_q}\}_{q<Q}$ into the dataset $\{X^{D,R,Q}, \text{Pc}^D\}$ where $\forall\ d < D,\ \forall\ q_0 < Q,\ \forall q_1 < Q,\ X^R_{d,q_0}$ and X^R_{d,q_1} result in the same operation, at the same iteration. The main constraints are the execution time and memory required. We proposed here a single-pass realignment algorithm detailed in *Schedule 5.1*. The fact that we only need the PC values to resynchronize the data significantly reduces the computational time and the needed memory. Indeed, our algorithm of resynchronization only have to read one time the $D_{q_{q<Q}}$ 64-bit PC values

instead of the $\{X^{D_q,R}\}_{q<Q}$ bit of data in the case of an algorithm based on the entire dataset. To provide details on the last assumption of the *Eq. 5.2*, the two *Def. 5.1* are needed. The whole algorithm described in the *Schedule 5.1* reveals the fix-points and the pseudo fix-points that are automatically realigned.

Definition 5.1. *Fix-point and pseudo fix-point*

A fix-point *is a PC value with a deterministic presence and a deterministic number of occurrence:*

$$\begin{aligned} &\mathrm{Pc}_d \in \{\mathrm{Pc}_q^{D_q}\}_{q<Q} \text{ is a fix-point} \\ &\iff \exists\ m \in \mathbb{N}, \sum_{d_q=1}^{D_q} \begin{cases} 1 & \text{if } \mathrm{Pc}_{d_q,q} = \mathrm{Pc}_d \\ 0 & \text{otherwise} \end{cases} = m\ \forall q < Q. \end{aligned}$$

- ***A pseudo fix-point*** *is a PC value with a deterministic number of appearance, in the case it appears (so a fix-point is also a pseudo fix-point):*

$$\begin{aligned} &\mathrm{Pc}_d \in \{\mathrm{Pc}_q^{D_q}\}_{q<Q} \text{ is a pseudo fix-point} \\ &\iff \exists\ m \in \mathbb{N}, \sum_{d_q=1}^{D_q} \begin{cases} 1 & \text{if } \mathrm{Pc}_{d_q,q} = \mathrm{Pc}_d \\ 0 & \text{otherwise} \end{cases} \in \{0, m\}\ \forall q < Q. \end{aligned}$$

To illustrate the proposed realignment algorithm, we first start with an application to a simple example. The *Fig. 5.3* displays a control flow graph with a conditional branching, a loop and a conditional branching inside. The letters A, B, C, ..., I are the PC values. The probability associated to each conditional branching are p_0 and p_1. The presented results have been obtained with $p_0 = 1/2$ and $p_1 = 1/3$. Three distinct executions of the proposed flow graph gave the following PC successions:

$$\{\mathrm{Pc}^{D_q}\}_{q<3} = \begin{cases} \mathrm{Pc}^{D_0} = A, B, D, E, H, I, E, H, I, E, F, G, I \\ \mathrm{Pc}^{D_1} = A, B, D, E, F, G, I, E, H, I, E, F, G, I \\ \mathrm{Pc}^{D_2} = A, C, D, E, H, I, E, H, I, E, F, G, I \end{cases} \tag{5.2}$$

The application of the proposed realignment algorithm, described in the *Schedule 5.1*, to a thousand executions of the flow graph detailed in *Fig. 5.3*, gives the matrix $\#\mathrm{Pc}^{D',Q}$ displayed in *Fig. 5.4*.

1. get the set of all possible PC values:

$$\mathrm{Pc}^{D'} = \bigcup_{q=0}^{Q-1} \mathrm{Pc}_q^{D_q}, \text{ with } D' \text{ the number of distinct values of PC over } Q$$

2. accumulate in the matrix $\#\mathrm{Pc}^{D',Q}$ the numbers of times each PC appears in each trace:

$$\#\mathrm{Pc}^{D',Q} = \left[\#\mathrm{Pc}_{d,q} = \sum_{d_q=0}^{D_q-1} \begin{cases} 1, & \text{if } \mathrm{Pc}_{d_q,q} = \mathrm{Pc}_d \\ 0, & \text{otherwise} \end{cases} \right]_{\substack{d<D' \\ q<Q}}$$

3. the fix-points and the pseudo fix-points are stored in F^D with its associated number of appearance:

$$\mathrm{F}^{D'',2} = \{(\mathrm{Pc}_d, m_d) \in (\mathrm{Pc}^{D'}, \mathbb{N}) | \exists\ m \in \mathbb{N},\ \#\mathrm{Pc}_{d,q} \in \{0, m\},\ \forall\ q < Q, \}$$

4. finally the axis $\mathrm{Pc}^{D(=\sum_{d=1}^{D''} m_d)}$ of PC used for the alignment is created using $\mathrm{F}^{D'',2}$:

$$\begin{aligned} \mathrm{Pc}^D &= \{\overbrace{\mathrm{F}_{0,0}, \ldots, \mathrm{F}_{0,0}}^{\mathrm{F}_{0,1} times}, \ldots, \overbrace{\mathrm{F}_{D''-1,0}, \ldots, \mathrm{F}_{D''-1,0}}^{\mathrm{F}_{D''-1,1} times}\} \\ &= \{\overbrace{\mathrm{Pc}_0, \ldots, \mathrm{Pc}_0}^{m_0 times}, \ldots, \overbrace{\mathrm{Pc}_{D''-1}, \ldots, \mathrm{Pc}_{D''-1}}^{m_{D''-1} times}\} \end{aligned}$$

Schedule 5.1: Step-by-step description of the realignment algorithm.

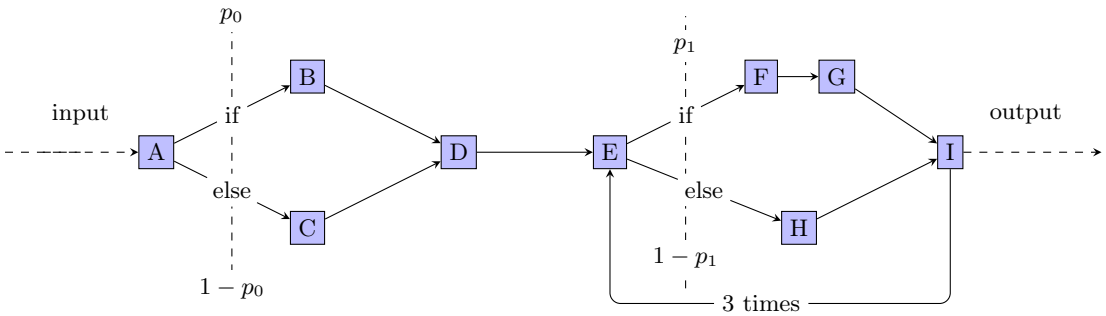


Figure 5.3: Control flow of the example.

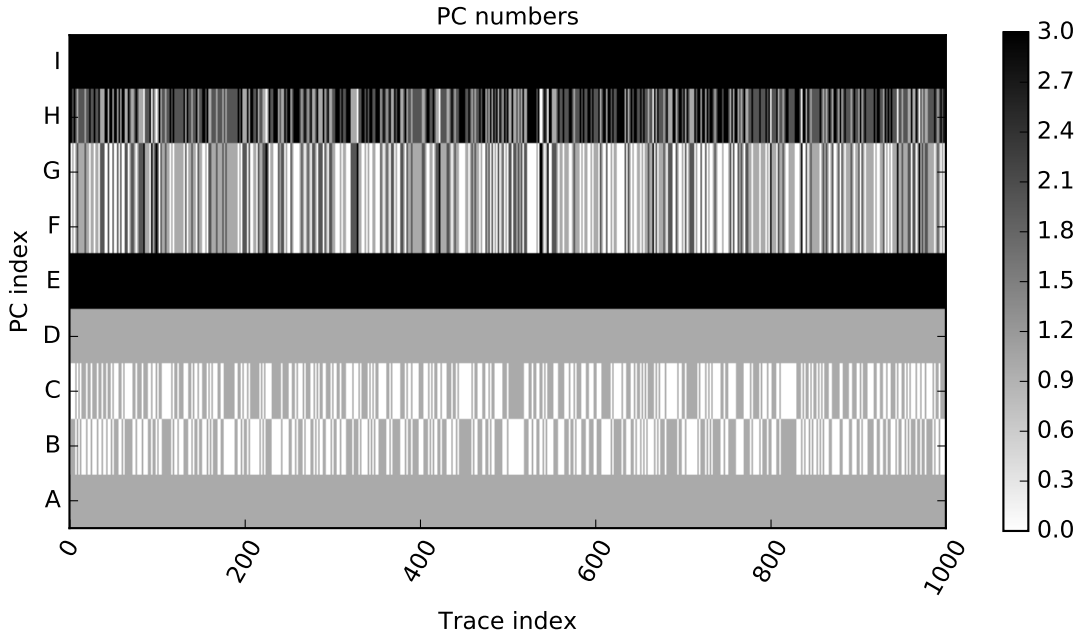


Figure 5.4: $\#\mathrm{Pc}^{D',Q}$ for $Q = 1000$ executions of the example described in *Fig. 5.3*, with $D' = 8$ and $\mathrm{Pc}^{D'} = \{A, B, C, D, \ldots, I\}$. The colors refer to the number of time each PC appear in each execution.

The realignment algorithm designates A, D, E and I as fix-points; B, C as pseudo fix-points while the PC F, G and H could not be realigned in the preliminary study. Those PC need more information to be realigned. Finally, the realignment algorithm gives the following output:

$$\mathrm{F}^{D''} = \{(A,1),(B,1),(C,1),(D,1),(E,3),(I,3)\}$$
$$\mathrm{Pc}^{D} = \{A, B, C, D, E, E, E, I, I, I\}$$

If we go back to the three execution traces given in the *Eq. 5.2*, the resynchronization is done as written in *Eq. 5.3*, with in red the elements affected by the realignment.

$$\{X_q^{D,R}\}_{q<3} =$$

$$\begin{cases} X_0^{D,R} = \{X_{0,0}^R, X_{0,1}^R, 0, X_{0,2}^R, X_{0,3}^R, X_{0,6}^R, X_{0,9}^R, X_{0,5}^R, X_{0,8}^R, X_{0,12}^R\} \\ X_1^{D,R} = \{X_{1,0}^R, X_{1,1}^R, 0, X_{1,2}^R, X_{1,3}^R, X_{1,7}^R, X_{1,10}^R, X_{1,6}^R, X_{1,9}^R, X_{1,13}^R\} \\ X_2^{D,R} = \{X_{2,0}^R, 0, X_{2,1}^R, X_{2,2}^R, X_{2,3}^R, X_{2,6}^R, X_{2,9}^R, X_{2,5}^R, X_{2,8}^R, X_{2,12}^R\} \end{cases} \tag{5.3}$$

The algorithm of realignment have been applied to a real-world masked implementation of an AES-128 , the results obtained are displayed in *Fig. 5.6*. It reveals that some PC, around the index 100 seem to be neither fix-points nor pseudo fix-points. This aspect is confirmed by the results of the computation of the mean and the standard deviation for the non-zero elements of $\#\mathrm{Pc}^{D',Q}$ over Q displayed in *Fig. 5.6*. In fact, this three figures show that two PC values have non-constant number of apparition in all the executions. In addition to give the PC that causes the misalignment and the resynchronized axis of PC, the algorithm helps to find the lines in the source code that provoke the misalignment. This matching from the PC values to the source lines code is made easier by the usage of GDB to record the leakage. Furthermore, the localization of the misalignment origins helps to identify timing leakage. In our example illustrated in *Fig. 5.6*, the realignment reveals that the misalignment is caused by the function *xtime* transcribed in *Fig. 5.5a*. This function multiplies the input b by two in $\mathbb{F}_{2^8}$, but this implementation contains conditional statement that misalign the data and that could produce time leakage. A possible improvement can be the usage of the constant time implementation of *xtime* provided in *Fig. 5.5b*. Our realignment algorithm immediately and precisely identify the non-constant time line in the source code. This information is very useful for a developer that want to implement constant time algorithm aim to protect his code against the *timing attacks*.

5.2.3 Data Reduction

Once the recorded data are realigned, if necessary, we have now access to the resynchronized data $X^{D,R,Q}$ and the associated PC vector Pc^D. Now that it is possible to analyze vertically the data, two questions arise. Are all the data in $X^{D,R,Q}$ relevant? And is it

```
unsigned char xtime (unsigned char b)
{
  unsigned char tmp = b<<1;
  return (b&0x80)?(tmp^0x1b) : tmp;
}
```

(a) implementation with conditional branch

```
unsigned char xtime (unsigned char b)
{
  return ((x<<1)^((x>>7)*0x1b));
}
```

(b) constant time implementation

Figure 5.5: Xtime implementations used in the mixColumn function of the AES

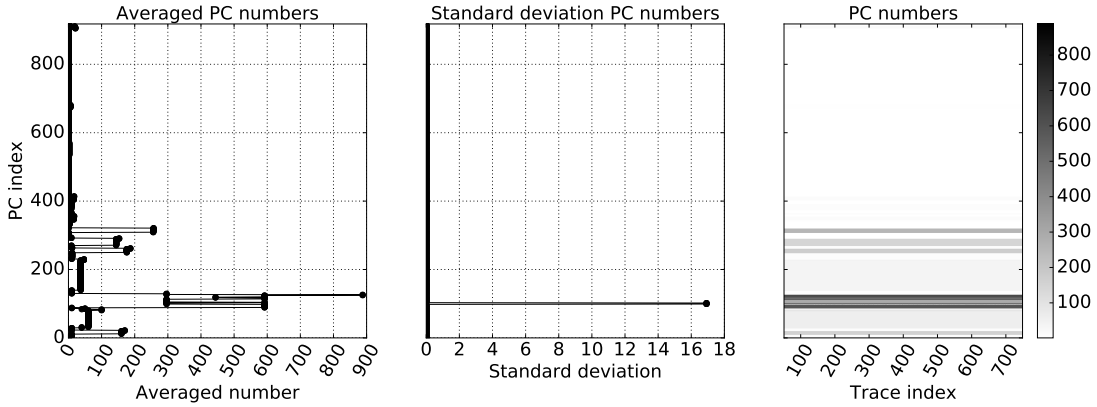


Figure 5.6: Results of the realignment algorithm applied to a masked implementation of an AES for thousand executions ($Q = 1000$). The mean is plotted on the left, the standard deviation over Q in the middle and $\#\mathrm{Pc}^{D',Q}$ in the right. The colors refer to the number of time each PC appear in each trace.

possible to reduce the dimension of the data to analyze? To answer these questions, we define the notion of activity matrix in *Def. 5.2* and of transition matrix *Def. 5.3*.

Definition 5.2. ***The activity matrix*** $A^{D,R}$ *of a given set* $X^{D,R,Q}$ *is defined as follow:*

$$A^{D,R} = \left[A_{d,r} = \begin{cases} 1, & \text{if } \sum_{q=0}^{Q-1} X_{d,r,q} \notin \{Q, 0\} \\ 0, & \text{otherwise} \end{cases} \right]_{\substack{d<D,\\ r<R}}$$

The activity matrix of a given dataset identifies the points $((d, r) \in D \times R)$ with a non-zero variance over Q.

Definition 5.3. ***The transition matrix*** $T^{D,R}$ *of a given set* $X^{D,R,Q}$ *is defined as follow:*

$$T^{D,R} = \left[T_{d,r} = \begin{cases} 1, & \text{if } d = 0 \\ \vee_{q=1}^{Q} X_{d-1,r} \oplus X_{d,r} & \text{otherwise} \end{cases} \right]_{\substack{d<D, \\ r<R}}$$

The transition matrix identifies the points $((d, r) \in D \times R)$ that change at least one time between the PC d and $d+1$ over all the traces. Both matrix $T^{D,R}$ and $A^{D,Q}$ could be computed in-line accumulating each trace. Then, we identify the points that could leak information as $L = \{(d, r) \in (D \times R) | T_{d,r} \wedge A_{d,r} = 1\}$. In *Fig. 5.7*, the matrices $A^{D,R}$, $T^{D,R}$ and $A^{D,R} \wedge T^{D,R}$ obtain analyzing a data set of 250 traces of execution of an AES-128 WBC implementation. We observe that our algorithm permits to identify the 0.29% from the entire samples that could leak information. Thus we conserve only 20190 PC values over 28277 and 120 bit register over 1472. This data reduction speeds up the analysis and reduce the memory footprint analyzing only the data $X^{L,Q}$. Furthermore we take advantage of the very low density of the sparse matrices $\{X_q^{D,R}\}_{q<Q}$ to reduce the storage required for the traces. The storage of sparse matrices is a well study problematic in computer science and a lot of solutions are freely provided. The following *Tab. 5.1* summarizes the gain in storage that we obtain using a method called *Compressed Sparse Column matrix (CSC)* present in *scipy*[1]. Thus, the needed memory to store the traces $X^{D,R,Q}$ decrease from $9,7$Go to 132Mo using the CSC compression on the matrix $T^{D,R,Q}$ defined in *Eq. 5.4*.

5.2.4 Distinguisher: CPA

The potential leakage points have been identified and stored in the dataset $X^{L,Q}$. To know if some of those points leak sensitive information, we use the CPA proposed by Brier *et al.* in 2004 [11]. In our case, the guessed intermediate values are stored in $Y^{S,Q,K,B}$ as explained in the *Eq. 5.1*. The CPA is based on the computation of the Pearson coefficient between $X^{L,Q}$ and $Y^{S,Q,K,B}$ to discriminate the right key $\{k^\star{}_b\}_{b<B}$

[1] https://www.scipy.org/

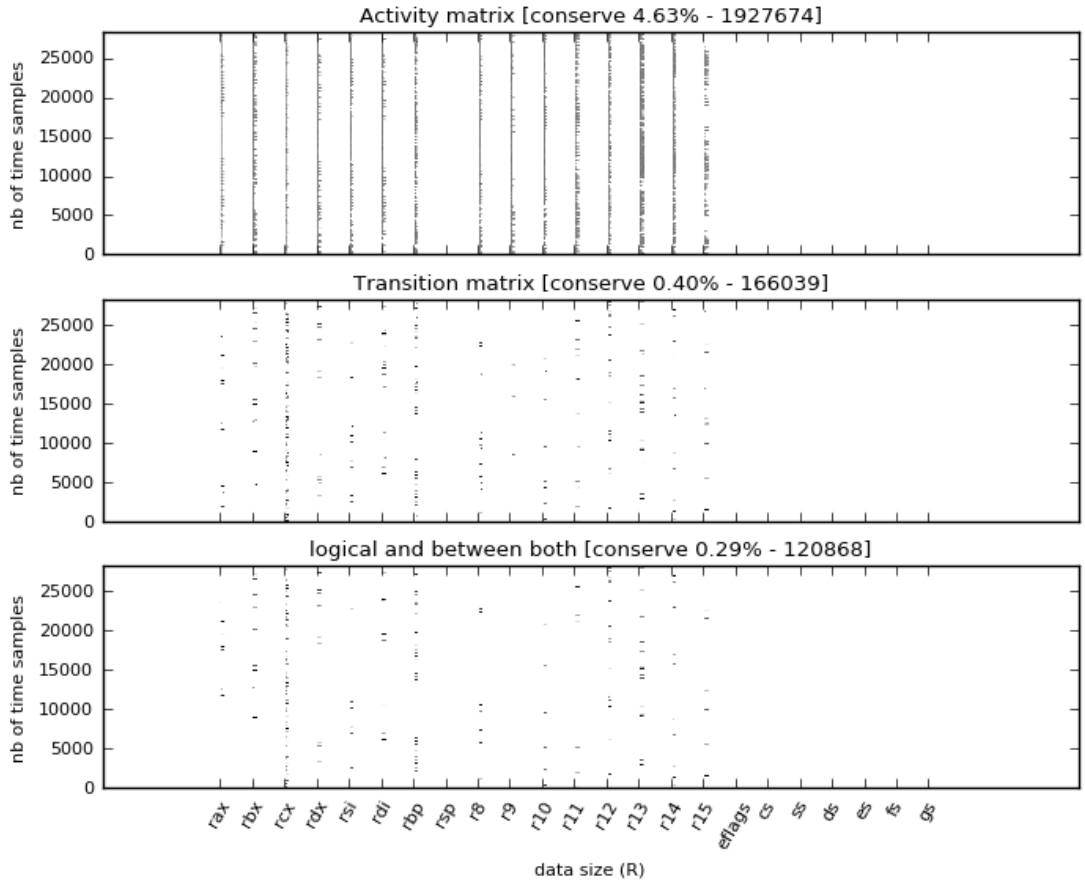


Figure 5.7: Illustration of the matrices $A^{D,R}$ (on the top),$T^{D,R}$ (in the middle) and $A^{D,R} \wedge T^{D,R}$ (on the bottom). In white are plotted the 0 and the 1 in black.

	$\{(X_q^{D,R}\}_{q<Q}$ 8-bit matrix	$\{(X_q^{D,R}\}_{q<Q}$ CSC 8-bit matrix	$\{(T_q^{D,R}\}_{q<Q}$ CSC 8-bit matrix
size	$9,7$Go	$2,6$Go	132Mo

with

$$\{(T_q^{D,R}\}_{q<Q} = \left\{ \begin{cases} X_{d,r,q}, & \text{if } d = 0 \\ X_{d-1,r,q} \oplus X_{d,r,q} & \text{otherwise} \end{cases} \right\}_{\substack{q<Q, r<R \\ d<D}} \tag{5.4}$$

Table 5.1: Benchmark of the needed memory to store the traces $X^{D,R,Q}$ for $Q = 250$ using distinct formats.

AbsMarg $(\mathcal{D}(X^{L,Q}, Y^{Q}_{s,b}))$

$$= \frac{\mathcal{D}(X^{L,Q}, Y^{Q}_{s,k^\star{}_b,b}) - \max_{\substack{k<256,\\ k\neq k^\star{}_b}}\{\mathcal{D}(X^{L,Q}, Y^{Q}_{s,k,b})\}}{\underbrace{\mathcal{D}(Y^{Q}_{s,k^\star{}_b,b}, Y^{Q}_{s,k^\star{}_b,b})}_{=1} - \max_{\substack{k<256,\\ k\neq k^\star{}_b}}\{\mathcal{D}(Y^{Q}_{s,k^\star{}_b,b}, Y^{Q}_{s,k,b})\}} \tag{5.6}$$

as defined in the following *Eq. 5.5*.

$$\mathcal{D}(X^{L,Q}, Y^{S,Q,K,B}) = \left\{ \frac{\mathrm{cov}(X^{L,Q}, Y^{Q}_{s,k,b})}{\sigma(X^{L,Q})\sigma(Y^{Q}_{s,k,b})} \right\}_{\substack{k<256, S<8\\ b<16}}, \tag{5.5}$$

where σ is the variance and cov the covariance both over Q.

Then we identify the leaking samples using the Absolute distinguishing Margin (AbsMarg) proposed by Whitnall *et al.* in [81] and recalled in the following *Eq. 5.6*. The usage of the AbsMarg metric is motivated by the presence of ghost peaks in the results of CPA. We take advantage that we analyze binary dataset to simplify the accumulative formula of the *Pearson coefficient* given in *Eq. 5.7*. This simplification speeds up the analysis and reduces the memory footprint.

$$\begin{aligned}
&\mathcal{D}(X^{L,Q}, Y^{Q}_{s,k,b}) \\
&\quad = \frac{\mathrm{acc_xy} - \frac{1}{Q}(\mathrm{acc_x.acc_y})}{\sqrt{\mathrm{acc_xx} - \frac{1}{Q}(\mathrm{acc_x.acc_x})}\sqrt{\mathrm{acc_yy} - \frac{1}{Q}(\mathrm{acc_y.acc_y})}} \\
&\quad = \frac{\mathrm{acc_xy} - \frac{1}{Q}(\mathrm{acc_x.acc_y})}{\sqrt{\mathrm{acc_x} - \frac{1}{Q}(\mathrm{acc_x.acc_x})}\sqrt{\mathrm{acc_y} - \frac{1}{Q}(\mathrm{acc_y.acc_y})}}
\end{aligned} \tag{5.7}$$

where we define the acc_ $\star$ as follow,

$$\begin{aligned}
\mathrm{acc_xy} &= \sum_{q=1}^{Q} X^{L}_{q}.Y_{s,k,b,q} \\
\mathrm{acc_x} &= \sum_{q=1}^{Q} X^{L}_{q} = \sum_{q=1}^{Q} (X^{L}_{q})^2 = \mathrm{acc_xx} \\
\mathrm{acc_y} &= \sum_{q=1}^{Q} Y_{s,k,b,q} = \sum_{q=1}^{Q} (Y_{s,k,b,q})^2 = \mathrm{acc_yy}
\end{aligned}$$

5.3 Results

In the following analyses, we use the known-plaintext attack model, which means that an attacker only requires access to the random inputs. Additionally, the attack could be performed with just the compiled binary. The source code access is only mandatory to map the leakages to the source code.

5.3.1 WBC Analysis

As explain in the previous *Subsec. 5.2.4* we choose the CPA to reveal leakage from $X^{L,Q}$ and discriminate the secret key $\{k^\star{}_b\}_{b<16}$ from the guesses $\{k_b | k_b \neq k^\star{}_b\}_{b<16}$. The *Fig. 5.8* illustrates the results obtained by applying the CPA on $Q = 250$ recorded traces from the WBC implementation provided at the CTF challenge of CHES-2016. The leakage model focuses the 8bit of the output of the S-box at the first, the corresponding Y is provided in *Formula 5.1*. The CPA has been computed for all samples L ($\sim$ 120k) but to facilitate the visualization we focus on the first 20k samples where the leaking ones are localized. In grey we display the results obtained with the bad guesses: $\left\{ \max_{k_b<256, \neq k^\star{}_b} (\mathcal{D}(X^{L,Q}, Y^Q_{s,k_b,b})) \right\}_{b<16,s<8}$ and in color the result obtained with the right key: $\left\{ \mathcal{D}(X^{L,Q}, Y^Q_{s,k^\star{}_b,b}) \right\}_{b<16,s<8}$. We observed a lot of colored peaks in the *Fig. 5.8* but if we compare with the AbsMarg results displayed in *Fig 5.9* only a small part of them really leak sensitive information. We apply a threshold at 0.25 to AbsMarg results to get the leaking samples.

Then, we take advantage of the PC values to map the identified leaking samples to the source code. We summarize the obtained results in the *Tab 5.2*. In this way, we accurately link each identified leak to a line code, a bit register and a leaking bit model.

Table 5.2: Leakage characterization and mapping to the source code for $S = 8$.

line	source code	R name	R bit	Y bit	key byte
l.4086:	v16 = lookup_nibble2(table_4436,v16,v18,0);	`rsi`	1	7	0
l.4420:	v18=lookup_nibble(table_13890,v4);	`r14`	3	4	1

l.4417: v2=lookup_nibble2(table_4934,v16,v2,0);	r13	3	4	1
l.4421: v19=lookup_nibble(table_13891,v4);	r13	2	4	1
l.4422: v16=lookup_nibble2(table_4940,v16,v18,0);	rcx	2	4	1
l.4199: v17=lookup_nibble2(table_4605,v17,v19,0);	r10	1	4	3
l.4200: v18=lookup_nibble(table_13732,v11);	rbx	1	4	3
l.4204: v18=lookup_nibble2(table_4611,v18,v20,0);	rbx	0	4	3
l.4204: v18=lookup_nibble2(table_4611,v18,v20,0);	rcx	0	4	3
l.4230: v28=lookup_nibble(table_13750,v12);	rbp	0	4	3
l.4227: v21=lookup_nibble2(table_4647,v21,v27,0);	rbx	2	0	3
l.4231: v29=lookup_nibble(table_13751,v12);	rcx	0	4	3
l.4232: v26=lookup_nibble2(table_4653,v26,v28,0);	rcx	2	4	3
l.4232: v26=lookup_nibble2(table_4653,v26,v28,0);	rbp	2	0	3
l.4232: v26=lookup_nibble2(table_4653,v26,v28,0);	rbp	1	0	3
l.4228: v26=lookup_nibble(table_13748,v11);	r12	4	4	3
l.4228: v26=lookup_nibble(table_13748,v11);	r12	5	4	3
l.4228: v26=lookup_nibble(table_13748,v11);	r12	6	4	3
l.4228: v26=lookup_nibble(table_13748,v11);	r12	7	4	3
l.4232: v26=lookup_nibble2(table_4653,v26,v28,0);	rcx	1	0	3
l.4233: v27=lookup_nibble2(table_4654,v27,v29,0);	rbx	4	4	3
l.4233: v27=lookup_nibble2(table_4654,v27,v29,0);	rbx	5	4	3
l.4233: v27=lookup_nibble2(table_4654,v27,v29,0);	rbx	6	4	3
l.4233: v27=lookup_nibble2(table_4654,v27,v29,0);	rbx	7	4	3
l.4232: v26=lookup_nibble2(table_4653,v26,v28,0);	rbx	3	4	3
l.4232: v26=lookup_nibble2(table_4653,v26,v28,0);	rbx	4	4	3
l.4232: v26=lookup_nibble2(table_4653,v26,v28,0);	rbx	5	4	3
l.4232: v26=lookup_nibble2(table_4653,v26,v28,0);	rbx	6	4	3
l.4233: v27=lookup_nibble2(table_4654,v27,v29,0);	rcx	3	4	3
l.4233: v27=lookup_nibble2(table_4654,v27,v29,0);	rcx	4	4	3
l.4233: v27=lookup_nibble2(table_4654,v27,v29,0);	rcx	5	4	3
l.4233: v27=lookup_nibble2(table_4654,v27,v29,0);	rcx	6	4	3
l.4198: v16=lookup_nibble2(table_4604,v16,v18,0);	rax	1	1	4
l.4198: v16=lookup_nibble2(table_4604,v16,v18,0);	rax	3	1	4

l.4328: v30=lookup_nibble(table_13822,v13);	rbp	3	3	7
l.4328: v30=lookup_nibble(table_13822,v13);	rbp	3	7	7
l.4329: v31=lookup_nibble(table_13823,v13);	rcx	3	3	7
l.4329: v31=lookup_nibble(table_13823,v13);	rcx	3	7	7
l.4329: v31=lookup_nibble(table_13823,v13);	rcx	2	3	7
l.4329: v31=lookup_nibble(table_13823,v13);	rcx	2	7	7
l.4309: v19=lookup_nibble(table_13811,v7);	rdx	0	2	8
l.4324: v18=lookup_nibble2(table_4793,v18,v20,0);	rbx	0	1	8
l.4091: v21=lookup_nibble(table_13655,v15);	rbx	1	7	10
l.4119: v25=lookup_nibble(table_13671,v15);	r12	0	3	10
l.4119: v25=lookup_nibble(table_13671,v15);	r12	1	0	10
l.4134: v5=lookup_nibble2(table_4506,v5,v23,0);	rbp	1	3	10
l.4465: v3=lookup_nibble2(table_5004,v3,v4,0);	r14	3	0	11
l.4470: v4=lookup_nibble2(table_5010,v4,v35,0);	rsi	3	0	11
l.4470: v4=lookup_nibble2(table_5010,v4,v35,0);	rsi	2	0	11
l.4470: v4=lookup_nibble2(table_5010,v4,v35,0);	rax	2	0	11
l.4436: v18=lookup_nibble2(table_4961,v18,v20,0);	rbx	0	2	12
l.4450: v20=lookup_nibble2(table_4982,v20,v34,0);	rbp	3	7	12
l.4463: v4=lookup_nibble(table_13915,v4);	r10	1	7	12
l.4217: v27=lookup_nibble(table_13743,v12);	r12	1	5	14

5.3.2 Analysis Improvement

As proposed in [2], we extend our leakage model Y provided in *Formula 5.1*, to take into account the two products computed during the mixColumn execution. In [2] authors proposed to compute three distinct 8-bit Differential Power Analysis (DPA) while the two products only add ten new bit-distributions to the initial model. Indeed the model extension makes growing the model size S from 8 to 18, and not to 24, because 6 bit-distributions are redundant as resumed in the following *Proposition 5.1*.

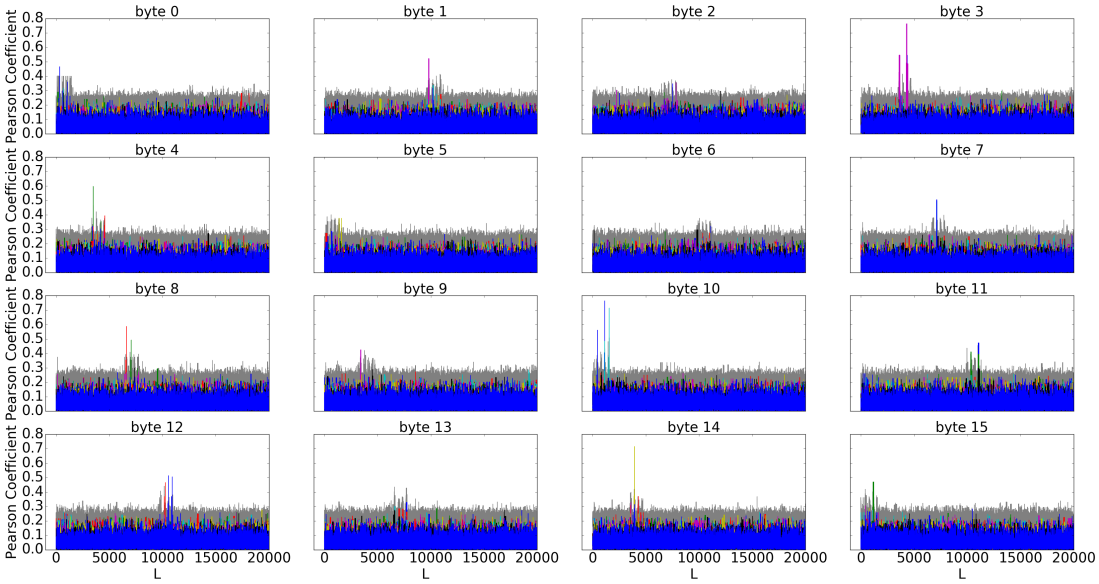


Figure 5.8: CPA, for $S = 8$, on recorded data from a WBC implementation.

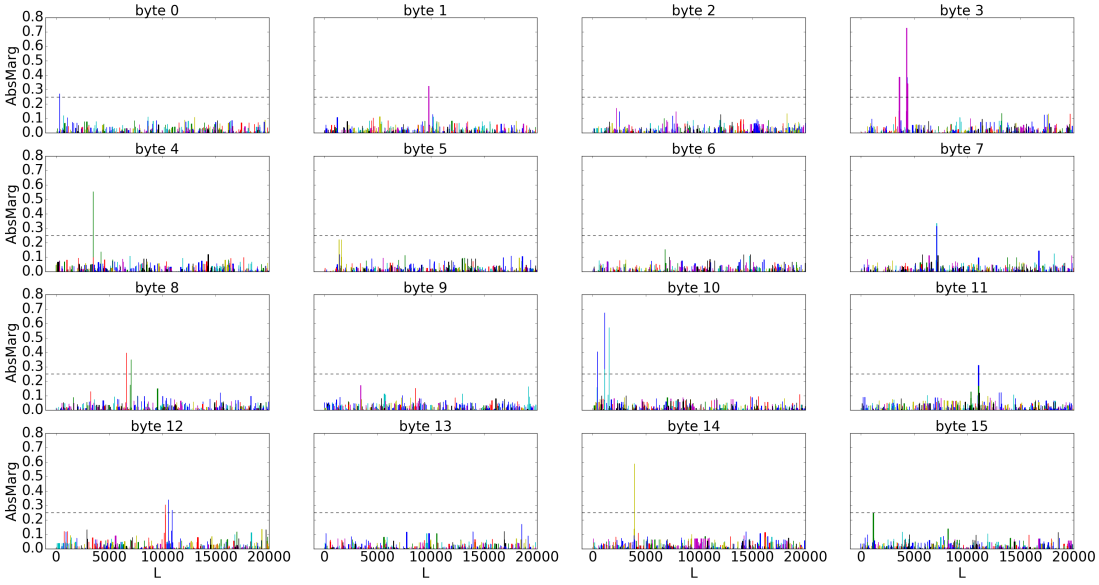


Figure 5.9: AbsMarg of the CPA for $S = 8$. Only the samples with a positive *AbsMarg* are plotted.

Proposition 5.1. $\forall x \in \mathbb{N}$, *with* $x < 256$ *the products by two and three computed in the mixColumn function respect the following properties:*

- *the bit* 2 *of* $2.x$ *is equal to the bit* 1 *of* x
- *the bit* 5 *of* $2.x$ *is equal to the bit* 4 *of* x
- *the bit* 6 *of* $2.x$ *is equal to the bit* 5 *of* x
- *the bit* 7 *of* $2.x$ *is equal to the bit* 6 *of* x
- *the bit* 0 *of* $2.x$ *is equal to the bit* 7 *of* x
- *the bit* 0 *of* $3.x$ *is equal to the bit* 1 *of* $2.x$

The *Proposition 5.1* permits to construct an extended model expressed in the following *formula 5.8*.

$$Y^{S,Q,K,B} =$$

$$\left[\begin{cases} \text{S-box}\ (T_{q,b} \oplus k)\ \&\ 2^s, & \text{if } s < 8 \\ (2.\text{S-box}\ (T_{q,b} \oplus k))\ \&\ 2^{v[s-8]}, & \text{if } 7 < s < 11, \text{ with } v = \{1,3,4\} \\ (3.\text{S-box}\ (T_{q,b} \oplus k))\ \&\ 2^{s-10}, & \text{if } 10 < s \end{cases}\right]_{\substack{s<18, k<256, \\ q<Q, b<16}} \tag{5.8}$$

As in the previous *Subsec. 5.3.1*, *Fig. 5.10* illustrates the results obtained by applying the CPA on $Q = 250$ recorded traces.In grey we display the results obtained with the bad guesses: $\left\{ \max_{k_b<256, k_b \neq k^\star_b} (\mathcal{D}(X^{L,Q}, Y^Q_{s,k_b,b})) \right\}_{b<16, s<18}$ and in color the result obtained with the right key: $\left\{ \mathcal{D}(X^{L,Q}, Y^Q_{s,k^\star_b,b}) \right\}_{b<16, s<18}$. The discrimination of

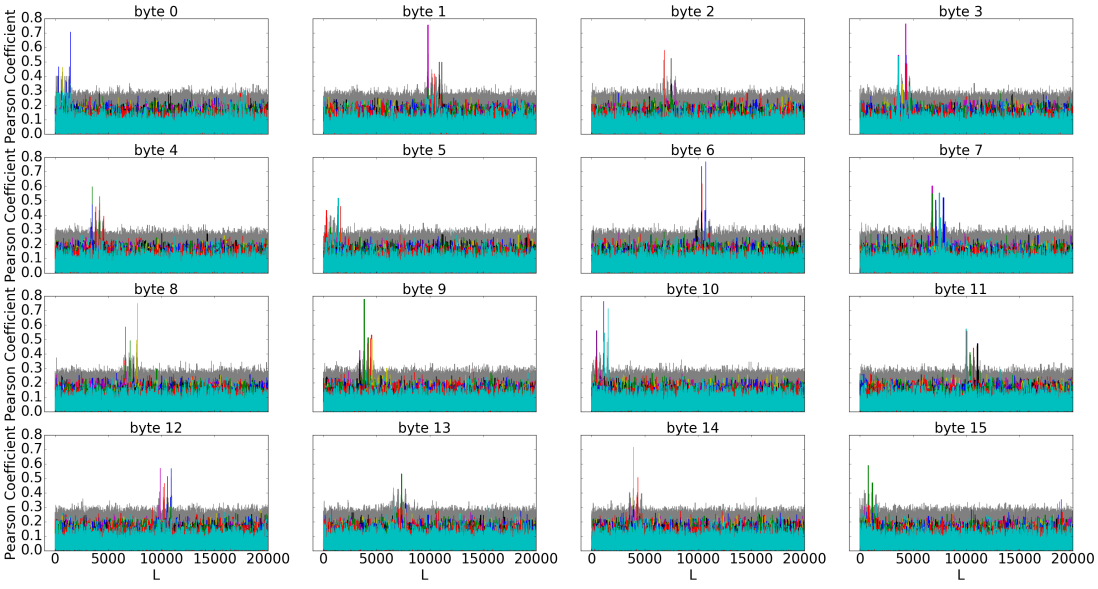


Figure 5.10: CPA, for $S = 18$, on recorded data from a WBC implementation.

the leaking samples from the computed CPA uses the AbsMarg displayed in *Fig 5.11*, in which we apply a threshold at 0.25.

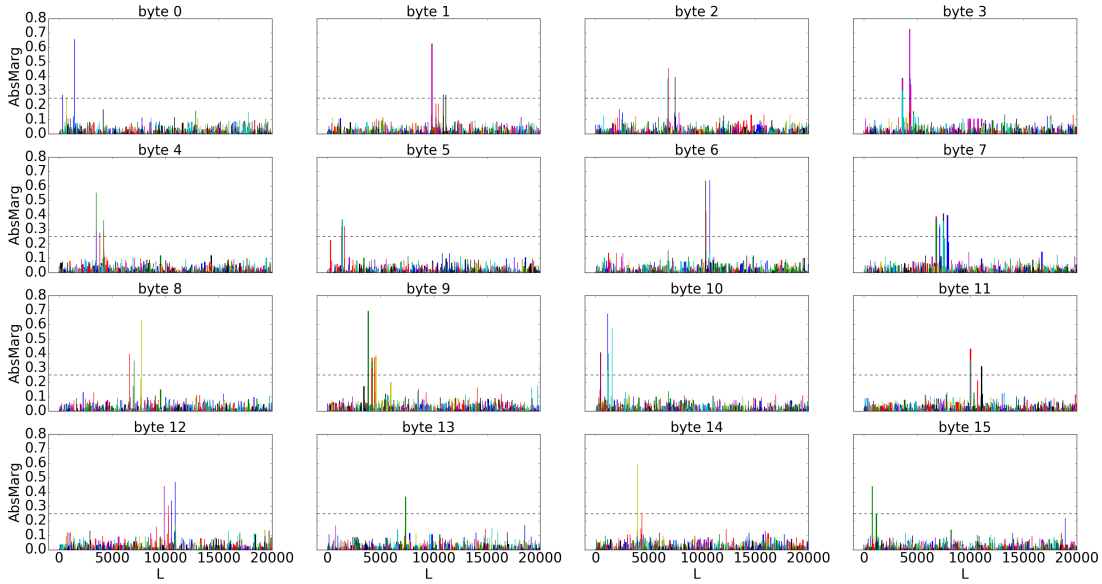


Figure 5.11: AbsMarg of the CPA for $S = 18$. Only the samples with a positive AbsMarg are plotted.

Then, using the PC values we map the identified leaking samples to the source code. We summarize the obtained results in the *Tab 5.3* in which we link each leakage to a line code, a bit register and a leaking bit model. First, the extended model lead us to recover the entire secret key $\{k^\star{}_b\}_{b<16}$ of the WBC implementation. Second the mapping of the leakage shows the diversity of the registers that could leak information: `rcx, rax, rbp, rdi, r8 r9, r10, r11, r13 r14, r15`; which justify the necessity to protect all the register of an hardware target. Finally, comparing both *Tabs 5.2* and 5.3 notify that a major part of the leakage results from the computing of the multiplication by three computed during the mixColumn execution.

Table 5.3: Leakage characterization and mapping to the source code for $S = 18$.

line	source code	R name	R bit	Y value	Y bit	key byte
l.4086:	v16=lookup_nibble2(table_4436,v16,v18,0);	`rsi`	1	x	7	0
l.4128:	v22=lookup_nibble2(table_4499,v22,v23,0);	`rbp`	0	3.x	4	0
l.4420:	v18=lookup_nibble(table_13890,v4);	`r14`	3	x	4	1
l.4417:	v2=lookup_nibble2(table_4934,v16,v2,0);	`r15`	1	3.x	1	1
l.4417:	v2=lookup_nibble2(table_4934,v16,v2,0);	`r13`	3	x	4	1

l.4421: v19=lookup_nibble(table_13891,v4);	r13	2	x	4	1
l.4422: v16=lookup_nibble2(table_4940,v16,v18,0);	rcx	2	x	4	1
l.4421: v19=lookup_nibble(table_13891,v4);	r13	1	3.x	1	1
l.4421: v19=lookup_nibble(table_13891,v4);	r13	0	3.x	1	1
l.4418: v16=lookup_nibble(table_13888,v3);	rdx	0	3.x	1	1
l.4463: v4=lookup_nibble(table_13915,v4);	r9	0	3.x	3	1
l.4462: v35=lookup_nibble(table_13914,v4);	rdi	0	3.x	3	1
l.4464: v34=lookup_nibble2(table_5003,v34,v35,0);	rcx	2	3.x	3	1
l.4315: v21=lookup_nibble(table_13815,v13);	rbx	0	3.x	7	2
l.4316: v18=lookup_nibble2(table_4779,v18,v20,0);	r13	1	3.x	6	2
l.4343: v33=lookup_nibble(table_13831,v13);	r12	2	3.x	3	2
l.4343: v33=lookup_nibble(table_13831,v13);	r12	3	3.x	3	2
l.4199: v17=lookup_nibble2(table_4605,v17,v19,0);	r10	0	3.x	7	3
l.4199: v17=lookup_nibble2(table_4605,v17,v19,0);	r10	1	x	4	3
l.4199: v17=lookup_nibble2(table_4605,v17,v19,0);	r10	3	3.x	7	3
l.4200: v18=lookup_nibble(table_13732,v11);	rbx	0	3.x	7	3
l.4200: v18=lookup_nibble(table_13732,v11);	rbx	1	x	4	3
l.4200: v18=lookup_nibble(table_13732,v11);	rbx	3	3.x	7	3
l.4204: v18=lookup_nibble2(table_4611,v18,v20,0);	rbx	0	x	4	3
l.4204: v18=lookup_nibble2(table_4611,v18,v20,0);	rbx	2	3.x	7	3
l.4204: v18=lookup_nibble2(table_4611,v18,v20,0);	rcx	0	x	4	3
l.4204: v18=lookup_nibble2(table_4611,v18,v20,0);	rcx	2	3.x	7	3
l.4204: v18=lookup_nibble2(table_4611,v18,v20,0);	rcx	2	3.x	7	3
l.4230: v28=lookup_nibble(table_13750,v12);	rbp	0	x	4	3
l.4227: v21=lookup_nibble2(table_4647,v21,v27,0);	rbx	2	x	0	3
l.4231: v29=lookup_nibble(table_13751,v12);	rcx	0	x	4	3
l.4232: v26=lookup_nibble2(table_4653,v26,v28,0);	rcx	2	x	4	3
l.4232: v26=lookup_nibble2(table_4653,v26,v28,0);	rbp	2	x	0	3
l.4232: v26=lookup_nibble2(table_4653,v26,v28,0);	rbp	1	x	0	3
l.4228: v26=lookup_nibble(table_13748,v11);	r12	4	x	4	3
l.4228: v26=lookup_nibble(table_13748,v11);	r12	5	x	4	3
l.4228: v26=lookup_nibble(table_13748,v11);	r12	6	x	4	3

l.4228: v26=lookup_nibble(table_13748,v11);	r12	7	x	4	3
l.4232: v26=lookup_nibble2(table_4653,v26,v28,0);	rcx	1	x	0	3
l.4233: v27=lookup_nibble2(table_4654,v27,v29,0);	rbx	4	x	4	3
l.4233: v27=lookup_nibble2(table_4654,v27,v29,0);	rbx	5	x	4	3
l.4233: v27=lookup_nibble2(table_4654,v27,v29,0);	rbx	6	x	4	3
l.4233: v27=lookup_nibble2(table_4654,v27,v29,0);	rbx	7	x	4	3
l.4232: v26=lookup_nibble2(table_4653,v26,v28,0);	rbx	3	x	4	3
l.4232: v26=lookup_nibble2(table_4653,v26,v28,0);	rbx	4	x	4	3
l.4232: v26=lookup_nibble2(table_4653,v26,v28,0);	rbx	5	x	4	3
l.4232: v26=lookup_nibble2(table_4653,v26,v28,0);	rbx	6	x	4	3
l.4233: v27=lookup_nibble2(table_4654,v27,v29,0);	rcx	3	x	4	3
l.4233: v27=lookup_nibble2(table_4654,v27,v29,0);	rcx	4	x	4	3
l.4233: v27=lookup_nibble2(table_4654,v27,v29,0);	rcx	5	x	4	3
l.4233: v27=lookup_nibble2(table_4654,v27,v29,0);	rcx	6	x	4	3
l.4198: v16=lookup_nibble2(table_4604,v16,v18,0);	rax	1	x	1	4
l.4198: v16=lookup_nibble2(table_4604,v16,v18,0);	rax	1	3.x	4	4
l.4198: v16=lookup_nibble2(table_4604,v16,v18,0);	rax	3	x	1	4
l.4211: v21=lookup_nibble(table_13739,v6);	rbp	3	3.x	6	4
l.4225: v27=lookup_nibble(table_13747,v6);	rbp	2	3.x	5	4
l.4225: v27=lookup_nibble(table_13747,v6);	rbp	3	2.x	3	4
l.4127: v5=lookup_nibble(table_13675,v5);	r9	0	2.x	3	5
l.4126: v23=lookup_nibble(table_13674,v5);	rdi	0	2.x	3	5
l.4123: v21=lookup_nibble2(table_4493,v21,v23,0);	r8	2	3.x	7	5
l.4127: v5=lookup_nibble(table_13675,v5);	r14	2	3.x	7	5
l.4127: v5=lookup_nibble(table_13675,v5);	r14	1	3.x	7	5
l.4128: v22=lookup_nibble2(table_4499,v22,v23,0);	rcx	1	3.x	7	5
l.4127: v5=lookup_nibble(table_13675,v5);	rcx	2	2.x	3	5
l.4441: v35=lookup_nibble(table_13903,v14);	r12	2	3.x	3	6
l.4442: v20=lookup_nibble2(table_4968,v20,v34,0);	r15	1	2.x	3	6
l.4442: v20=lookup_nibble2(table_4968,v20,v34,0);	r15	3	2.x	3	6
l.4456: v34=lookup_nibble2(table_4989,v34,v36,0);	r15	3	3.x	4	6
l.4311: v17=lookup_nibble2(table_4773,v17,v19,0);	r10	1	3.x	5	7

l.4311: v17=lookup_nibble2(table_4773,v17,v19,0);	r10	2	3.x	1	7
l.4312: v18=lookup_nibble(table_13812,v8);	rbx	1	3.x	5	7
l.4312: v18=lookup_nibble(table_13812,v8);	rbx	2	3.x	1	7
l.4316: v18=lookup_nibble2(table_4779,v18,v20,0);	rbx	0	3.x	5	7
l.4316: v18=lookup_nibble2(table_4779,v18,v20,0);	rbx	1	3.x	1	7
l.4316: v18=lookup_nibble2(table_4779,v18,v20,0);	rcx	0	3.x	5	7
l.4316: v18=lookup_nibble2(table_4779,v18,v20,0);	rcx	1	3.x	1	7
l.4328: v30=lookup_nibble(table_13822,v13);	rbp	3	x	3	7
l.4328: v30=lookup_nibble(table_13822,v13);	rbp	3	x	7	7
l.4329: v31=lookup_nibble(table_13823,v13);	rcx	3	x	3	7
l.4329: v31=lookup_nibble(table_13823,v13);	rcx	3	x	7	7
l.4329: v31=lookup_nibble(table_13823,v13);	rcx	2	x	3	7
l.4329: v31=lookup_nibble(table_13823,v13);	rcx	2	x	7	7
l.4342: v32=lookup_nibble(table_13830,v13);	rbp	0	3.x	7	7
l.4342: v32=lookup_nibble(table_13830,v13);	rbp	1	3.x	3	7
l.4343: v33=lookup_nibble(table_13831,v13);	rcx	0	3.x	7	7
l.4343: v33=lookup_nibble(table_13831,v13);	rcx	1	3.x	3	7
l.4343: v33=lookup_nibble(table_13831,v13);	rcx	0	3.x	3	7
l.4344: v30=lookup_nibble2(table_4821,v30,v32,0);	rcx	2	3.x	7	7
l.4357: v13=lookup_nibble(table_13839,v13);	r14	0	3.x	4	7
l.4358: v7=lookup_nibble2(table_4842,v7,v31,0);	r13	0	3.x	4	7
l.4359: v8=lookup_nibble2(table_4843,v8,v13,0);	rcx	2	3.x	4	7
l.4309: v19=lookup_nibble(table_13811,v7);	rdx	0	x	2	8
l.4324: v18=lookup_nibble2(table_4793,v18,v20,0);	rbx	0	x	1	8
l.4352: v30=lookup_nibble2(table_4835,v30,v31,0);	rbp	0	3.x	2	8
l.4352: v30=lookup_nibble2(table_4835,v30,v31,0);	rbp	3	3.x	2	8
l.4207: v17=lookup_nibble2(table_4619,v17,v19,0);	r10	2	3.x	5	9
l.4208: v18=lookup_nibble(table_13736,v1);	rbp	2	3.x	5	9
l.4208: v18=lookup_nibble(table_13736,v1);	rbp	1	3.x	5	9
l.4212: v18=lookup_nibble2(table_4625,v18,v20,0);	rcx	1	3.x	5	9
l.4225: v27=lookup_nibble(table_13747,v6);	r11	2	3.x	6	9
l.4224: v26=lookup_nibble(table_13746,v6);	r14	2	3.x	6	9

l.4221: v19=lookup_nibble2(table_4640,v19,v21,0);	r10	0	2.x	3	9
l.4221: v19=lookup_nibble2(table_4640,v19,v21,0);	r10	2	3.x	5	9
l.4221: v19=lookup_nibble2(table_4640,v19,v21,0);	r14	1	3.x	6	9
l.4225: v27=lookup_nibble(table_13747,v6);	rcx	1	3.x	6	9
l.4222: v20=lookup_nibble(table_13744,v1);	rbp	0	2.x	3	9
l.4222: v20=lookup_nibble(table_13744,v1);	rbp	2	3.x	5	9
l.4225: v27=lookup_nibble(table_13747,v6);	rbp	1	3.x	5	9
l.4226: v20=lookup_nibble2(table_4646,v20,v26,0);	rcx	1	3.x	5	9
l.4234: v20=lookup_nibble2(table_4660,v20,v26,0);	rcx	2	2.x	3	9
l.4238: v27=lookup_nibble(table_13754,v6);	r9	3	2.x	3	9
l.4239: v6=lookup_nibble(table_13755,v6);	r8	2	3.x	2	9
l.4235: v21=lookup_nibble2(table_4661,v21,v27,0);	rdi	3	2.x	3	9
l.4240: v26=lookup_nibble2(table_4667,v26,v27,0);	rdi	2	2.x	3	9
l.4239: v6=lookup_nibble(table_13755,v6);	r15	2	3.x	2	9
l.4239: v6=lookup_nibble(table_13755,v6);	rcx	2	2.x	3	9
l.4239: v6=lookup_nibble(table_13755,v6);	r15	1	3.x	2	9
l.4241: v1=lookup_nibble2(table_4668,v1,v6,0);	rcx	1	3.x	2	9
l.4091: v21=lookup_nibble(table_13655,v15);	rbx	1	x	7	10
l.4092: v18=lookup_nibble2(table_4443,v18,v20,0);	r13	1	3.x	6	10
l.4119: v25=lookup_nibble(table_13671,v15);	r12	0	x	3	10
l.4119: v25=lookup_nibble(table_13671,v15);	r12	1	x	0	10
l.4120: v22=lookup_nibble2(table_4485,v22,v24,0);	r15	0	2.x	4	10
l.4134: v5=lookup_nibble2(table_4506,v5,v23,0);	rbp	1	x	3	10
l.4426: v20=lookup_nibble(table_13894,v14);	r11	3	3.x	7	11
l.4423: v17=lookup_nibble2(table_4941,v17,v19,0);	r10	2	2.x	3	11
l.4427: v21=lookup_nibble(table_13895,v14);	rcx	3	3.x	7	11
l.4427: v21=lookup_nibble(table_13895,v14);	rcx	2	3.x	7	11
l.4424: v18=lookup_nibble(table_13892,v9);	rbx	2	2.x	3	11
l.4428: v18=lookup_nibble2(table_4947,v18,v20,0);	rbx	1	2.x	3	11
l.4428: v18=lookup_nibble2(table_4947,v18,v20,0);	rcx	1	2.x	3	11
l.4465: v3=lookup_nibble2(table_5004,v3,v4,0);	r14	3	x	0	11
l.4465: v3=lookup_nibble2(table_5004,v3,v4,0);	r14	3	3.x	3	11

l.4470: v4=lookup_nibble2(table_5010,v4,v35,0);	rsi	3	x	0	11
l.4470: v4=lookup_nibble2(table_5010,v4,v35,0);	rsi	3	3.x	3	11
l.4470: v4=lookup_nibble2(table_5010,v4,v35,0);	rsi	2	x	0	11
l.4470: v4=lookup_nibble2(table_5010,v4,v35,0);	rsi	2	3.x	3	11
l.4470: v4=lookup_nibble2(table_5010,v4,v35,0);	rax	2	x	0	11
l.4470: v4=lookup_nibble2(table_5010,v4,v35,0);	rax	2	3.x	3	11
l.4425: v19=lookup_nibble(table_13893,v9);	rax	1	3.x	1	12
l.4436: v18=lookup_nibble2(table_4961,v18,v20,0);	rbx	0	x	2	12
l.4450: v20=lookup_nibble2(table_4982,v20,v34,0);	rbp	3	x	7	12
l.4463: v4=lookup_nibble(table_13915,v4);	r10	0	3.x	4	12
l.4463: v4=lookup_nibble(table_13915,v4);	r10	1	x	7	12
l.4337: v31=lookup_nibble(table_13827,v7);	r11	2	2.x	1	13
l.4336: v30=lookup_nibble(table_13826,v7);	r14	2	2.x	1	13
l.4333: v19=lookup_nibble2(table_4808,v19,v21,0);	r14	1	2.x	1	13
l.4337: v31=lookup_nibble(table_13827,v7);	rcx	1	2.x	1	13
l.4217: v27=lookup_nibble(table_13743,v12);	r12	1	x	5	14
l.4104: v22=lookup_nibble(table_13662,v15);	rbp	1	3.x	5	15
l.4105: v23=lookup_nibble(table_13663,v15);	rcx	1	3.x	5	15
l.4105: v23=lookup_nibble(table_13663,v15);	rcx	0	3.x	5	15

5.4 Conclusion

We present in this paper a new methodology of SC evaluation for software implementation. We push the state of the art in that field by providing a practical and effective methodology to extract all the data that will be manipulated by a software implementation during its execution. All the recorded data are analyzed independently, at bit level, without any leakage model applied to generate traces as in all the SC simulators presented in the state of the art (as far as we know). These features give to our solution exhaustive properties and suppress the noise that a leakage model could generate. Our exhaustive approach makes it agnostic for the target hardware by focusing the analysis on the manipulated data and not on hardware characteristics. Furthermore, our

methodology allows to map leakage of sensitive information to the source code, and this can be of significant help for an evaluator or a developer. Advantageously, we provide two additional new methods to support and improve the SCA assessment. First, we describe an efficient resynchronization algorithm based on the control flow values. Second, we give a methodology to significantly decrease the number of samples to analyze. Both features are crucial when analyzing complex and/or massive software implementations. Furthermore, our solution could be plugged in others data providers than GDB. As example, we applied our methodology to data provided by the *Virtulyzr*[26] to analyze hardware implementations. In the case, the PC are replaced by time values and the registers by wires.

CHAPTER 6

Conclusion and Perspectives

6.1 Conclusion

Since the explosion of the market connected devices, security became a major preoccupation across domains: military, civil, medical... Every day new threats and security breaches are revealed. The Side-Channel (SC) domain is not spared, even more with the blast of the Internet of Thing (IoT). This thesis is in line with this growing need of security increasing the understanding of the leakage. Indeed, understanding the nature of the leakage, in terms of model complexity and leaking samples, is vital to construct countermeasures relevantly.

First, we provide close forms of optimal distinguisher in the context of multivariate leakage and multivariate models. These results are recalled in the two following *Fig. 6.1* and 6.2. All the formula gathered there permit to maximize the probability of success in many cases. Thus we provide the optimal distinguishers close forms for distinct profiles of attacker:

- the perfect attacker: the model (α, Σ) is perfectly known,
- the intermediate attacker: the model has been computed in a learning set,

- the weak attacker: the attacker has only access to the data set under attack.

Additionally, we treat the influence of the noise, studying the autoregressive and the isotropic noise distributions. This complete study gives us a clear methodology to exploit all the leaking information whichever its complexity. Furthermore, we treat the computational aspect giving fast implementation of the optimal distinguisher.

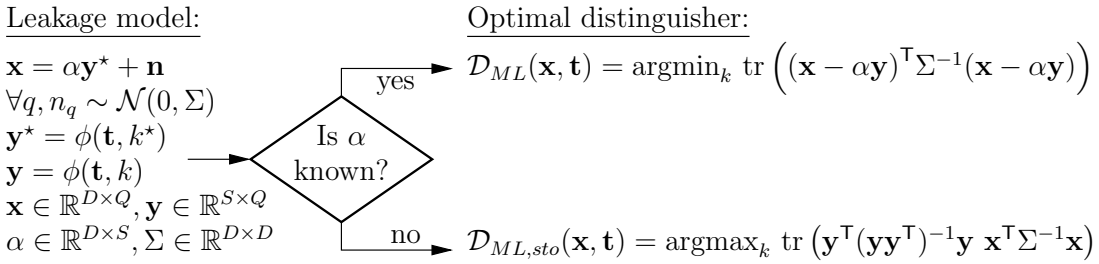


Figure 6.1: Mathematical expression for multivariate ($D \geq 1$) optimal attacks with a linear combination of models ($S \geq 1$)

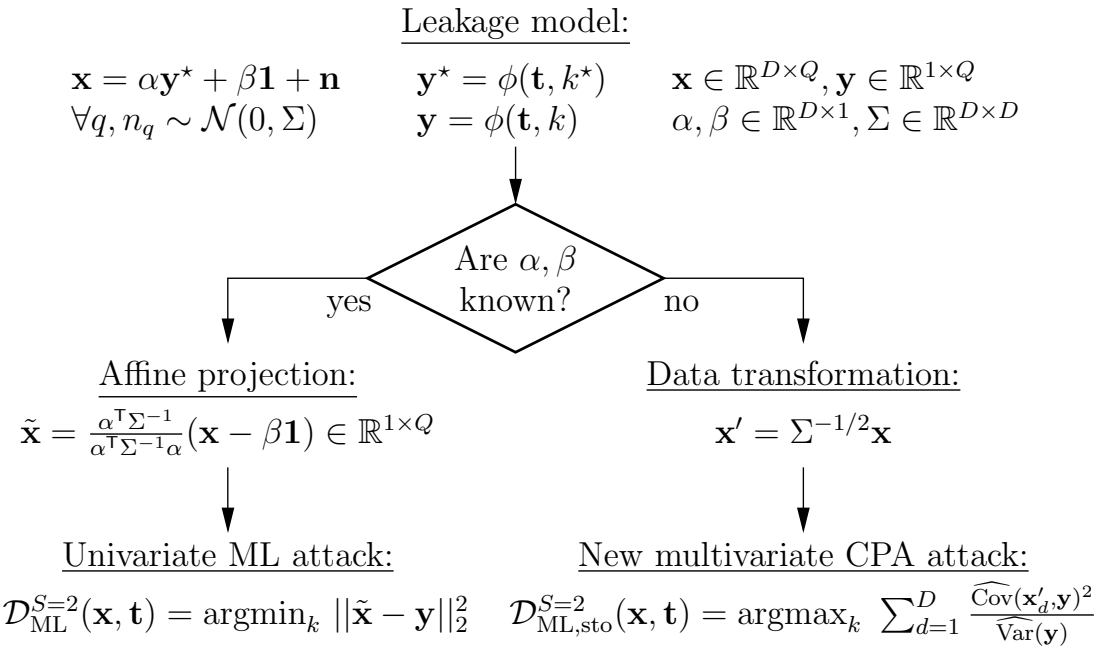


Figure 6.2: Modus operandi for multivariate ($D \geq 1$) optimal attacks with one model Y associated to envelope $\alpha \in \mathbb{R}^D$ and a constant offset $\beta \in \mathbb{R}^D$ ($S = 2$)

Second, we provide a complete framework to collect and analyze data provided by GNU Debugger (GDB). We exhaustively analyze and understand the leakage that could result

from the execution of a software implementation. Additionally, we efficiently solve the two vital problems of resynchronization and of denoising. This *end-to-end* process starts with a source code implementation and finishes with a mapping leakage to the source code. The relation between the source code and the leakage is a powerful information. Thus, a software developer can quickly identify and understand the leakage that arises from his code.

6.2 Perspectives

Linking an optimal analysis of physical leakage resulting from the executing of a software code and a software analysis with GDB of the same implementation should be an interesting perspective. The results that we would get should improve the understanding on the nature of the leakage. Another perspective should be a comparison between the optimal distinguishers we invented [13, 15] and the new distinguishers based on neural network or machine learning [17, 50]. Improving the data collection should be an inspiring perspective for the software analysis. GDB allows accurate gathering of data. However, GDB is slow, in that it provides a huge amount of features we do not need for our application case. Alternative methods for instance leveraging `ptrace` (internally used by GDB) can be both faster and easier portable, albeit at the cost of a reduced number of features. Speeding up the data collection will give us the opportunity to analyze bigger codes, not only cryptographic algorithms but complete protocols. Another improvement of the proposed methodology of software analysis should be the application of the optimal distinguishers to the collected data. Furthermore taking advantage of the binary aspect of the multidimensional dataset $X^{D,\mathbb{R},Q}$ and the multidimensional model ($Y^{S,Q}$) could lead to an improvement of distinguishing step. In fact, as for the Correlation Power Analysis (CPA) (*Formula 5.7*), the binary format can help to simplify the formula. Lastly, the usage of a "bitmap-format implementation"[1] could speed-up the computation and decrease the memory footprint distributions discrimination. Another axis of study is to find an efficient method to exhaustively combine leaking samples to reveal potential high order leakage over our huge binary dataset. Finally, as discussed during the reviewing process, investigations and improvements have to be found to enable the analysis of codes that embed polymorphic countermeasure [22]. At present, our algorithm of resynchronization is beaten by polymorphic codes. A way to analyze such

[1] As provided by the python library https://pypi.org/project/bitmap/

codes is to take advantage of additional information as the executer bytecofr values.

Bibliography

[1] Specification for the Advanced Encryption Standard (AES). Federal Information Processing Standards Publication 197, 2001.
→ Cited on page 26.

[2] Hyunjin Ahn and Dong-Guk Han. Multilateral white-box cryptanalysis: Case study on WB-AES of CHES challenge 2016. *IACR Cryptology ePrint Archive*, 2016:807, 2016.
→ Cited on pages 85 and 99.

[3] Julien Allibert, Benoit Feix, Georges Gagnerot, Ismael Kane, Hugues Thiebeauld, and Tiana Razafindralambo. Chicken or the egg - computational data attacks or physical attacks. *IACR Cryptology ePrint Archive*, 2015:1086, 2015.
→ Cited on page 84.

[4] ANSSI (Agence Nationale de la Sécurité des Systèmes d'Information). Mécanismes cryptographiques - Règles et recommandations, 2014.
→ Cited on page 29.

[5] Cédric Archambeau, Éric Peeters, François-Xavier Standaert, and Jean-Jacques Quisquater. Template Attacks in Principal Subspaces. In *CHES*, volume 4249 of *LNCS*, pages 1–14. Springer, October 10-13 2006. Yokohama, Japan.
→ Cited on pages 45, 46, 56, and 57.

[6] Elaine B. Barker, William C. Barker, William E. Burr, W. Timothy Polk, and Miles E. Smid. Sp 800-57. recommendation for key management, part 1: General (revised), 2007.
→ Cited on page 29.

[7] Lejla Batina, Jip Hogenboom, and Jasper G. J. van Woudenberg. Getting more from PCA: first results of using principal component analysis for extensive power analysis. In Orr Dunkelman, editor, *Topics in Cryptology - CT-RSA 2012 - The Cryptographers' Track at the RSA Conference 2012, San Francisco, CA, USA, February 27 - March 2, 2012. Proceedings*, volume 7178 of *Lecture Notes in Computer Science*, pages 383–397. Springer, 2012.
→ Cited on page 46.

[8] Shivam Bhasin, Jean-Luc Danger, Sylvain Guilley, and Zakaria Najm. Side-channel Leakage and Trace Compression Using Normalized Inter-class Variance. In *Proceedings of the Third Workshop on Hardware and Architectural Support for Security and Privacy*, HASP '14, pages 7:1–7:9, New York, NY, USA, 2014. ACM.
→ Cited on page 46.

[9] Joppe W. Bos, Charles Hubain, Wil Michiels, and Philippe Teuwen. Differential computation analysis: Hiding your white-box designs is not enough. In Benedikt Gierlichs and Axel Y. Poschmann, editors, *Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*, volume 9813 of *Lecture Notes in Computer Science*, pages 215–236. Springer, 2016.
→ Cited on pages 84 and 85.

[10] Antoine Bouvet, Nicolas Bruneau, Adrien Facon, Sylvain Guilley, and Damien Marion. Give Me Your Binary, I'll Tell You If It Leaks. In *13th International Conference on Design & Technology of Integrated Systems In Nanoscale Era, DTIS 2018, Taormina, Italy, April 10-12, 2018*, 2018.
→ Cited on pages 18 and 43.

[11] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2004.
→ Cited on pages 34, 45, 68, 88, and 94.

[12] Nicolas Bruneau, Jean-Luc Danger, Sylvain Guilley, Annelie Heuser, and Yannick Teglia. Boosting Higher-Order Correlation Attacks by Dimensionality Reduction. In Rajat Subhra Chakraborty, Vashek Matyas, and Patrick Schaumont, editors, *Security, Privacy, and Applied Cryptography Engineering - 4th International Conference, SPACE 2014, Pune, India, October 18-22, 2014. Proceedings*, volume 8804 of *Lecture Notes in Computer Science*, pages 183–200. Springer, 2014.
→ Cited on pages 32 and 47.

[13] Nicolas Bruneau, Sylvain Guilley, Annelie Heuser, Damien Marion, and Olivier Rioul. Less is more - dimensionality reduction from a theoretical perspective. In Tim Güneysu and Helena Handschuh, editors, *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*, volume 9293 of *Lecture Notes in Computer Science*, pages 22–41. Springer, 2015.
→ Cited on pages 17, 32, 42, 67, 72, and 111.

[14] Nicolas Bruneau, Sylvain Guilley, Annelie Heuser, Damien Marion, and Olivier Rioul. Optimal side-channel attacks for multivariate leakages and multiple models. In *Security Proofs for Embedded Systemss - PROOF 2016 - 5th workshop edition, Santa Barbara, USA, August 20, 2016, Proceedings*, 2016.
→ Cited on pages 17 and 43.

[15] Nicolas Bruneau, Sylvain Guilley, Annelie Heuser, Damien Marion, and Olivier Rioul. Optimal side-channel attacks for multivariate leakages and multiple models. *J. Cryptographic Engineering*, 7(4):331–341, 2017.
→ Cited on pages 17, 43, and 111.

[16] BSI (Bundesamt für Sicherheit in der Informationstechnik). Kryptographische Verfahren: Empfehlungen und Schlüssellängen, 2017.
→ Cited on page 29.

[17] Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. Convolutional neural networks with data augmentation against jitter-based countermeasures - profiling attacks without pre-processing. In *CHES*, volume 10529 of *Lecture Notes in Computer Science*, pages 45–68. Springer, 2017.
→ Cited on page 111.

[18] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware*

and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer, 2002.
→ Cited on pages 34, 45, 46, 47, 66, and 68.

[19] Christophe Clavier. An improved SCARE cryptanalysis against a secret A3/A8 GSM algorithm. In *ICISS*, volume 4812 of *Lecture Notes in Computer Science*, pages 143–155. Springer, 2007.
→ Cited on page 29.

[20] Christophe Clavier, Jean-Luc Danger, Guillaume Duc, M. Abdelaziz Elaabid, Benoît Gérard, Sylvain Guilley, Annelie Heuser, Michael Kasper, Yang Li, Victor Lomné, Daisuke Nakatsu, Kazuo Ohta, Kazuo Sakiyama, Laurent Sauvage, Werner Schindler, Marc Stöttinger, Nicolas Veyrat-Charvillon, Matthieu Walle, and Antoine Wurcker. Practical improvements of side-channel attacks on AES: feedback from the 2nd DPA contest. *Journal of Cryptographic Engineering*, pages 1–16, 2014.
→ Cited on page 61.

[21] Christophe Clavier, Quentin Isorez, Damien Marion, and Antoine Wurcker. Complete reverse-engineering of AES-like block ciphers by SCARE and FIRE attacks. *Cryptography and Communications*, 7(1):121–162, 2015.
→ Cited on page 29.

[22] Damien Couroussé, Thierno Barry, Bruno Robisson, Nicolas Belleville, Philippe Jaillon, Olivier Potin, Hélène Le Bouder, Jean-Louis Lanet, and Karine Heydemann. All paths lead to rome: Polymorphic runtime code generation for embedded systems. In John Goodacre, Mikel Luján, Giovanni Agosta, Alessandro Barenghi, Israel Koren, and Gerardo Pelosi, editors, *Proceedings of the Fifth Workshop on Cryptography and Security in Computing Systems, CS2 2018, Manchester, United Kingdom, January 24, 2018*, pages 17–18. ACM, 2018.
→ Cited on page 111.

[23] Thomas M. Cover and Joy A. Thomas. *Elements of information theory (2. ed.)*. Wiley, 2006.
→ Cited on page 47.

[24] Guillaume Dabosville, Julien Doget, and Emmanuel Prouff. A new second-order side channel attack based on linear regression. *IEEE Trans. Computers*, 62(8):1629–

1640, 2013.
→ Cited on page 82.

[25] Jean-Luc Danger, Nicolas Debande, Sylvain Guilley, and Youssef Souissi. High-order timing attacks. In *Proceedings of the First Workshop on Cryptography and Security in Computing Systems*, CS2 '14, pages 7–12, New York, NY, USA, 2014. ACM.
→ Cited on page 46.

[26] Jean-Luc Danger, Sylvain Guilley, Philippe Nguyen, Robert Nguyen, and Youssef Souissi. Analyzing security breaches of countermeasures throughout the refinement process in hardware design flow. In David Atienza and Giorgio Di Natale, editors, *Design, Automation & Test in Europe Conference & Exhibition, DATE 2017, Lausanne, Switzerland, March 27-31, 2017*, pages 1129–1134. IEEE, 2017.
→ Cited on page 108.

[27] Nicolas Debande, Youssef Souissi, M. Abdelaziz Elaabid, Sylvain Guilley, and Jean-Luc Danger. Wavelet transform based pre-processing for side channel analysis. In *45th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 2012, Workshops Proceedings, Vancouver, BC, Canada, December 1-5, 2012*, pages 32–38. IEEE Computer Society, 2012.
→ Cited on page 46.

[28] Nicolas Debande, Youssef Souissi, Maxime Nassar, Sylvain Guilley, Thanh-Ha Le, and Jean-Luc Danger. "re-synchronization by moments": An efficient solution to align side-channel traces. In *2011 IEEE International Workshop on Information Forensics and Security, WIFS 2011, Iguacu Falls, Brazil, November 29 - December 2, 2011*, pages 1–6. IEEE Computer Society, 2011.
→ Cited on page 88.

[29] Julien Doget, Emmanuel Prouff, Matthieu Rivain, and François-Xavier Standaert. Univariate side channel attacks and leakage modeling. *J. Cryptographic Engineering*, 1(2):123–144, 2011.
→ Cited on page 88.

[30] François Durvaux, François-Xavier Standaert, Nicolas Veyrat-Charvillon, Jean-Baptiste Mairy, and Yves Deville. Efficient selection of time samples for higher-order DPA with projection pursuits. In Stefan Mangard and Axel Y. Poschmann, editors, *Constructive Side-Channel Analysis and Secure Design - 6th International*

Workshop, COSADE 2015, Berlin, Germany, April 13-14, 2015. Revised Selected Papers, volume 9064 of *Lecture Notes in Computer Science*, pages 34–50. Springer, 2015.
→ Cited on page 47.

[31] ECRYPT II. ECRYPT II Yearly Report on Algorithms and Keysizes, 2012.
→ Cited on page 29.

[32] Adrien Facon, Sylvain Guilley, Matthieu Lec'Hvien, Damien Marion, and Thomas Perianin. Binary data analysis for source code leakage assessment. In Jean-Louis Lanet and Cristian Toma, editors, *Innovative Security Solutions for Information Technology and Communications - 11th International Conference, SecITC 2018, Bucharest, Romania, November 8-9, 2018, Revised Selected Papers*, volume 11359 of *Lecture Notes in Computer Science*, pages 391–409. Springer, 2018.
→ Cited on pages 18 and 43.

[33] Yunsi Fei, Qiasi Luo, and A. Adam Ding. A statistical model for DPA with novel algorithmic confusion analysis. In Prouff and Schaumont [61], pages 233–250.
→ Cited on pages 50 and 51.

[34] Benedikt Gierlichs, Kerstin Lemke-Rust, and Christof Paar. Templates vs. stochastic methods. In Louis Goubin and Mitsuru Matsui, editors, *Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings*, volume 4249 of *Lecture Notes in Computer Science*, pages 15–29. Springer, 2006.
→ Cited on pages 45, 46, and 77.

[35] Cezary Glowacz, Vincent Grosso, Romain Poussier, Joachim Schüth, and François-Xavier Standaert. Simpler and more efficient rank estimation for side-channel security assessment. In Gregor Leander, editor, *Fast Software Encryption - 22nd International Workshop, FSE 2015, Istanbul, Turkey, March 8-11, 2015, Revised Selected Papers*, volume 9054 of *Lecture Notes in Computer Science*, pages 117–129. Springer, 2015.
→ Cited on page 67.

[36] Sylvain Guilley, Karim Khalfallah, Victor Lomné, and Jean-Luc Danger. Formal framework for the evaluation of waveform resynchronization algorithms. In Claudio Agostino Ardagna and Jianying Zhou, editors, *Information Security Theory and Practice. Security and Privacy of Mobile Devices in Wireless Communication - 5th*

IFIP WG 11.2 International Workshop, WISTP 2011, Heraklion, Crete, Greece, June 1-3, 2011. Proceedings, volume 6633 of *Lecture Notes in Computer Science*, pages 100–115. Springer, 2011.
→ Cited on page 88.

[37] Sylvain Guilley, Laurent Sauvage, Philippe Hoogvorst, Renaud Pacalet, Guido Marco Bertoni, and Sumanta Chaudhuri. Security evaluation of WDDL and seclib countermeasures against power attacks. *IEEE Trans. Computers*, 57(11):1482–1497, 2008.
→ Cited on page 46.

[38] Suvadeep Hajra and Debdeep Mukhopadhyay. Multivariate leakage model for improving non-profiling DPA on noisy power traces. In Dongdai Lin, Shouhuai Xu, and Moti Yung, editors, *Information Security and Cryptology - 9th International Conference, Inscrypt 2013, Guangzhou, China, November 27-30, 2013, Revised Selected Papers*, volume 8567 of *Lecture Notes in Computer Science*, pages 325–342. Springer, 2013.
→ Cited on page 47.

[39] Suvadeep Hajra and Debdeep Mukhopadhyay. SNR to success rate: Reaching the limit of non-profiling DPA. *IACR Cryptology ePrint Archive*, 2013:865, 2013.
→ Cited on page 47.

[40] Suvadeep Hajra and Debdeep Mukhopadhyay. On the optimal pre-processing for non-profiling differential power analysis. In Emmanuel Prouff, editor, *Constructive Side-Channel Analysis and Secure Design - 5th International Workshop, COSADE 2014, Paris, France, April 13-15, 2014. Revised Selected Papers*, volume 8622 of *Lecture Notes in Computer Science*, pages 161–178. Springer, 2014.
→ Cited on pages 45 and 47.

[41] Suvadeep Hajra and Debdeep Mukhopadhyay. Reaching the limit of nonprofiling DPA. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 34(6):915–927, 2015.
→ Cited on page 47.

[42] Annelie Heuser, Olivier Rioul, and Sylvain Guilley. Good is not good enough - deriving optimal distinguishers from communication theory. In Lejla Batina and Matthew Robshaw, editors, *Cryptographic Hardware and Embedded Systems - CHES 2014 - 16th International Workshop, Busan, South Korea, September 23-26,*

2014. Proceedings, volume 8731 of *Lecture Notes in Computer Science*, pages 55–74. Springer, 2014.
→ Cited on pages 17, 34, 47, 67, 68, 69, and 77.

[43] Jérémy Jean. TikZ for Cryptographers. https://www.iacr.org/authors/tikz/, 2016.
→ Cited on page 27.

[44] Peter Karsmakers, Benedikt Gierlichs, Kristiaan Pelckmans, Katrien De Cock, Johan Suykens, Bart Preneel, and Bart De Moor. Side channel attacks on cryptographic devices as a classification problem. COSIC technical report, 2009.
→ Cited on page 46.

[45] Auguste Kerckhoffs. *La Cryptographie Militaire*, volume IX of *Lecture Notes in Computer Science*. 1883.
→ Cited on page 29.

[46] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.
→ Cited on pages 15, 28, 34, and 48.

[47] Arjen K. Lenstra. Key length. contribution to the handbook of information security, 2004.
→ Cited on page 29.

[48] Arjen K. Lenstra and Eric R. Verheul. Selecting cryptographic key sizes. *J. Cryptology*, 14(4):255–293, 2001.
→ Cited on page 29.

[49] Victor Lomné, Emmanuel Prouff, and Thomas Roche. Behind the scene of side channel attacks. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology - ASIACRYPT 2013 - 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part I*, volume 8269 of *Lecture Notes in Computer Science*, pages 506–525. Springer, 2013.
→ Cited on pages 42 and 73.

[50] Houssem Maghrebi, Thibault Portigliatti, and Emmanuel Prouff. Breaking cryptographic implementations using deep learning techniques. In Claude Carlet, M. Anwar Hasan, and Vishal Saraswat, editors, *Security, Privacy, and Applied Cryptography Engineering - 6th International Conference, SPACE 2016, Hyderabad, India, December 14-18, 2016, Proceedings*, volume 10076 of *Lecture Notes in Computer Science*, pages 3–26. Springer, 2016.
→ Cited on page 111.

[51] Prasanta Chandra Mahalanobis. On the Generalised Distance in Statistics. *Proceedings of the National Institute of Sciences of India*, 2(1):49—55, 1936.
→ Cited on page 69.

[52] Luke Mather, Elisabeth Oswald, and Carolyn Whitnall. Multi-target DPA attacks: Pushing DPA beyond the limits of a desktop computer. In Sarkar and Iwata [65], pages 243–261.
→ Cited on page 67.

[53] David McCann, Carolyn Whitnall, and Elisabeth Oswald. ELMO: emulating leaks for the ARM cortex-m0 without access to a side channel lab. *IACR Cryptology ePrint Archive*, 2016:517, 2016.
→ Cited on page 84.

[54] Nicolas Debande and Maël Berthier and Yves Bocktaels and Thanh-Ha Le. Profiled Model Based Power Simulator for Side Channel Evaluation. *IACR Cryptology ePrint Archive*, 2012:703, 2012.
→ Cited on page 84.

[55] Roman Novak. Side-channel attack on substitution blocks. In *ACNS*, volume 2846 of *Lecture Notes in Computer Science*, pages 307–318. Springer, 2003.
→ Cited on page 29.

[56] NSA (National Security Agency). Commercial National Security Algorithm, Information Assurance Directorate, 2014.
→ Cited on page 29.

[57] Markus G. Kuhn Omar Choudary. Efficient template attacks. In Aurélien Francillon and Pankaj Rohatgi, editors, *Smart Card Research and Advanced Applications - 12th International Conference, CARDIS 2013, Berlin, Germany, November 27-29, 2013. Revised Selected Papers*, volume 8419 of *Lecture Notes in Computer Science*,

pages 253–270. Springer, 2013.
→ Cited on pages 46 and 69.

[58] David Oswald and Christof Paar. Improving side-channel analysis with optimal linear transforms. In Stefan Mangard, editor, *Smart Card Research and Advanced Applications - 11th International Conference, CARDIS 2012, Graz, Austria, November 28-30, 2012, Revised Selected Papers*, volume 7771 of *Lecture Notes in Computer Science*, pages 219–233. Springer, 2012.
→ Cited on pages 45 and 46.

[59] Elisabeth Oswald, Stefan Mangard, Christoph Herbst, and Stefan Tillich. Practical Second-Order DPA Attacks for Masked Smart Card Implementations of Block Ciphers. In David Pointcheval, editor, *CT-RSA*, volume 3860 of *Lecture Notes in Computer Science*, pages 192–207. Springer, 2006.
→ Cited on page 46.

[60] Emmanuel Prouff, Matthieu Rivain, and Régis Bevan. Statistical analysis of second order differential power analysis. *IEEE Trans. Computers*, 58(6):799–811, 2009.
→ Cited on page 39.

[61] Emmanuel Prouff and Patrick Schaumont, editors. *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings*, volume 7428 of *Lecture Notes in Computer Science*. Springer, 2012.
→ Cited on pages 118 and 122.

[62] Mathieu Renauld, François-Xavier Standaert, Nicolas Veyrat-Charvillon, Dina Kamel, and Denis Flandre. A formal study of power variability issues and side-channel attacks for nanoscale devices. In Kenneth G. Paterson, editor, *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, volume 6632 of *Lecture Notes in Computer Science*, pages 109–128. Springer, 2011.
→ Cited on page 46.

[63] Oscar Reparaz, Benedikt Gierlichs, and Ingrid Verbauwhede. Selecting time samples for multivariate DPA attacks. In Prouff and Schaumont [61], pages 155–174.
→ Cited on page 47.

[64] Matthieu Rivain and Thomas Roche. SCARE of secret ciphers with SPN structures. In *ASIACRYPT (1)*, volume 8269 of *Lecture Notes in Computer Science*, pages 526–544. Springer, 2013.
→ Cited on page 29.

[65] Palash Sarkar and Tetsu Iwata, editors. *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, volume 8873 of *Lecture Notes in Computer Science*. Springer, 2014.
→ Cited on pages 121 and 125.

[66] Akashi Satoh. Side-channel Attack Standard Evaluation Board, SASEBO-GII. Project of the AIST – RCIS (Research Center for Information Security), http://www.rcis.aist.go.jp/special/SASEBO/SASEBO-GII-en.html [Accessed on May 31, 2015].
→ Cited on page 61.

[67] Werner Schindler, Kerstin Lemke, and Christof Paar. A stochastic model for differential side channel cryptanalysis. In Josyula R. Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, volume 3659 of *Lecture Notes in Computer Science*, pages 30–46. Springer, 2005.
→ Cited on page 74.

[68] Youssef Souissi, Nicolas Debande, Sami Mekki, Sylvain Guilley, Ali Maalaoui, and Jean-Luc Danger. On the Optimality of Correlation Power Attack on Embedded Cryptographic Systems. In Ioannis G. Askoxylakis, Henrich Christopher Pöhls, and Joachim Posegga, editors, *WISTP*, volume 7322 of *Lecture Notes in Computer Science*, pages 169–178. Springer, June 20-22 2012.
→ Cited on page 46.

[69] Youssef Souissi, Maxime Nassar, Sylvain Guilley, Jean-Luc Danger, and Florent Flament. First principal components analysis: A new side channel distinguisher. In Kyung Hyune Rhee and DaeHun Nyang, editors, *Information Security and Cryptology - ICISC 2010 - 13th International Conference, Seoul, Korea, December 1-3, 2010, Revised Selected Papers*, volume 6829 of *Lecture Notes in Computer Science*, pages 407–419. Springer, 2010.
→ Cited on page 46.

[70] François-Xavier Standaert and Cédric Archambeau. Using Subspace-Based Template Attacks to Compare and Combine Power and Electromagnetic Information Leakages. In *CHES*, volume 5154 of *Lecture Notes in Computer Science*, pages 411–425. Springer, August 10–13 2008. Washington, D.C., USA.
→ Cited on pages 46 and 60.

[71] Daehyun Strobel, David Oswald, Bastian Richter, Falk Schellenberg, and Christof Paar. Microcontrollers as (in)security devices for pervasive computing applications. *Proceedings of the IEEE*, 102(8):1157–1173, 2014.
→ Cited on page 46.

[72] Takeshi Sugawara, Naofumi Homma, Takafumi Aoki, and Akashi Satoh. Profiling attack using multivariate regression analysis. *IEICE Electronic Express*, 7(15):1139–1144, 2010.
→ Cited on pages 46, 51, and 67.

[73] TELECOM ParisTech. DPA Contest, 2nd edition. http://www.DPAcontest.org/v2/ [Accessed on May 31, 2015].
→ Cited on pages 9, 19, 20, 45, 61, and 64.

[74] TELECOM ParisTech SEN research group. DPA Contest (4th edition), 2013–2014. http://www.DPAcontest.org/v4/.
→ Cited on pages 11, 16, 21, 33, 34, and 80.

[75] Hugues Thiebeauld, Georges Gagnerot, Antoine Wurcker, and Christophe Clavier. Scatter : A new dimension in side-channel. Cryptology ePrint Archive, Report 2017/706, 2017. https://eprint.iacr.org/2017/706.
→ Cited on page 88.

[76] Céline Thuillet, Philippe Andouard, and Olivier Ly. A smart card power analysis simulator. In *Proceedings of the 12th IEEE International Conference on Computational Science and Engineering, CSE 2009, Vancouver, BC, Canada, August 29-31, 2009*, pages 847–852. IEEE Computer Society, 2009.
→ Cited on page 84.

[77] Jasper G. J. van Woudenberg, Marc F. Witteman, and Bram Bakker. Improving differential power analysis by elastic alignment. In Aggelos Kiayias, editor, *Topics in Cryptology - CT-RSA 2011 - The Cryptographers' Track at the RSA Conference 2011, San Francisco, CA, USA, February 14-18, 2011. Proceedings*, volume 6558

of *Lecture Notes in Computer Science*, pages 104–119. Springer, 2011.
→ Cited on page 88.

[78] Nikita Veshchikov. SILK: high level of abstraction leakage simulator for side channel analysis. In Mila Dalla Preda and Jeffrey Todd McDonald, editors, *Proceedings of the 4th Program Protection and Reverse Engineering Workshop, PPREW@ACSAC 2014, New Orleans, LA, USA, December 9, 2014*, pages 3:1–3:11. ACM, 2014.
→ Cited on page 84.

[79] Nicolas Veyrat-Charvillon, Benoît Gérard, Mathieu Renauld, and François-Xavier Standaert. An optimal key enumeration algorithm and its application to side-channel attacks. In Lars R. Knudsen and Huapeng Wu, editors, *Selected Areas in Cryptography, 19th International Conference, SAC 2012, Windsor, ON, Canada, August 15-16, 2012, Revised Selected Papers*, volume 7707 of *Lecture Notes in Computer Science*, pages 390–406. Springer, 2012.
→ Cited on page 67.

[80] Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. Soft analytical side-channel attacks. In Sarkar and Iwata [65], pages 282–296.
→ Cited on page 67.

[81] Carolyn Whitnall and Elisabeth Oswald. A fair evaluation framework for comparing side-channel distinguishers. *J. Cryptographic Engineering*, 1(2):145–160, 2011.
→ Cited on pages 6 and 96.

Multidimensionality of the Models and the Data in the Side-Channel Domain

Damien Marion

RÉSUMÉ : Depuis la publication en 1999 du papier fondateur de Paul C. Kocher, Joshua Jaffe et Benjamin Jun, intitulé "Differential Power Analysis", les attaques par canaux auxiliaires se sont révélées être un moyen d'attaque performant contre les algorithmes cryptographiques. En effet, il s'est avéré que l'utilisation d'information extraite de canaux auxiliaires comme le temps d'exécution, la consommation de courant ou les émanations électromagnétiques, pouvait être utilisée pour retrouver des clés secrètes.

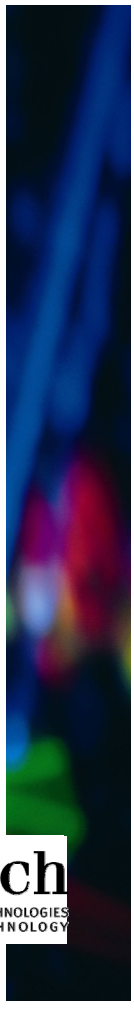
C'est dans ce contexte que cette thèse propose, dans un premier temps, de traiter le problème de la réduction de dimension. En effet, en vingt ans, la complexité ainsi que la taille des données extraites des canaux auxiliaires n'a cessé de croître. C'est pourquoi la réduction de dimension de ces données permet de réduire le temps et d'augmenter l'efficacité des attaques. Les méthodes de réduction de dimension proposées le sont pour des modèles de fuites complexe et de dimension quelconques. Dans un second temps, une méthode d'évaluation d'algorithmes logiciels est proposée. Celle-ci repose sur l'analyse de l'ensemble des données manipulées lors de l'exécution du logiciel évalué. La méthode proposée est composée de plusieurs fonctionnalités permettant d'accélérer et d'augmenter l'efficacité de l'analyse, notamment dans le contexte d'évaluation d'implémentation de cryptographie en boîte blanche.

MOTS-CLEFS : Attaques par canaux auxiliaires, réduction de dimension, modèles de fuites, cryptographie en boîte blanche, AES.

ABSTRACT : Since the publication in 1999 of the seminal paper of Paul C. Kocher, Joshua Jaffe and Benjamin Jun, entitled "Differential Power Analysis", the side-channel attacks have been proved to be efficient ways to attack cryptographic algorithms. Indeed, it has been revealed that the usage of information extracted from the side-channels such as the execution time, the power consumption or the electromagnetic emanations could be used to recover secret keys.

In this context, we propose first, to treat the problem of dimensionality reduction. Indeed, since twenty years, the complexity and the size of the data extracted from the side-channels do not stop to grow. That is why the reduction of these data decreases the time and increases the efficiency of these attacks. The dimension reduction is proposed for complex leakage models and any dimension. Second, a software leakage assessment methodology is proposed ; it is based on the analysis of all the manipulated data during the execution of the software. The proposed methodology provides features that speed-up and increase the efficiency of the analysis, especially in the case of white box cryptography.

KEY-WORDS : Side-channel attacks, dimensionality reduction, leakage models, white box cryptography, AES.



TELECOM ParisTech

EDiTE

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY