Analyse et Développement Logiciel

Gaétan BOIS–BAUMANN
Anaïs ESPICIER
Noah STAPLE

Encadrant: **Nicolas Aragon**

Master Cryptis
Université de Limoges

10 *Janvier* 2025

Contents

Attaques par canaux auxiliaires et contre-mesures pour les chiffrements AES et RSA

Keywords: Attaques par canaux auxiliaires, AES, RSA, Attaques par analyse de consommation électrique, Chipwhisperer, Hiding, Masking, SPA, DPA, CPA, Rétro-ingénierie AES

1 Introduction

Les attaques par canaux auxiliaires ou *(Side-Channel Attacks ou **SCA**)* sont des attaques passives et non-invasives où l'attaquant va observer et mesurer les caractéristiques analogiques de l'implémentation logicielle ou matérielle d'un algorithme de chiffrement, afin d'extraire la clé de chiffrement. Les principales caractéristiques analogiques utilisées en SCA sont le **temps d'exécution**, la **consommation électrique** et les **émissions électromagnétiques**. [2] [3].

Dans le cadre de ce projet, nous nous pencherons sur les attaques visant les algorithmes symétriques **AES** et asymétriques **RSA**, en analysant leur **consommation d'énergie** pendant l'exécution.

Après avoir analysé et mis en œuvre les différentes attaques, nous nous pencherons sur les diverses contre-mesures suggérées dans la littérature scientifique pour y faire face.

1.1 Advanced Encryption Standard (AES)

Advanced Encryption Standard (AES) [4, 5] est un standard décrivant un algorithme cryptographique permettant de protéger des données électroniques. AES a été approuvé par le National Institute of Standards and Technology (NIST) le 26 Novembre 2001.

Cet algorithme est un algorithme de chiffrement symétrique par blocs, dérivé de la famille d'algorithmes de Rijndael, qui utilise différentes tailles de clés et de blocs. AES est utilisé en 3 versions différentes, AES-128, AES-192 et AES-256 qui utilisent respectivement une taille de clé de 128, 192 et 256 bits ainsi qu'une taille de blocs de 128 bits.

Fonctionemment

AES transforme un texte clair P en un texte chiffré C. Pour réaliser cette transformation, AES étant dérivé de Rijndael, son principe de fonctionnement est similaire. Le processus de chiffrement est basé sur des rondes. Le nombre de rondes dépend de la taille de la clé:

- 10 rondes pour une clé de 128 bits
- 12 rondes pour une clé de 192 bits
- 14 rondes pour une clé de 256 bits

À chaque ronde, une clé distincte, nommée **Round Key**, est utilisée, et toutes ces clés sont dérivées de la clé secrète SK.

Une ronde comprend plusieurs étapes, que nous allons détailler ci-dessous.

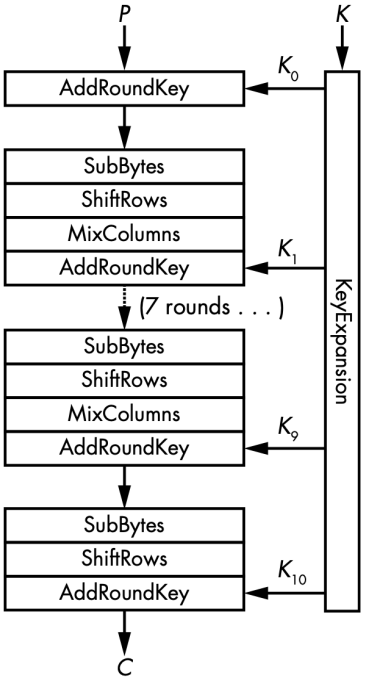


Figure 1: Rondes AES-128 [6]

Expansion de la clé *(Key Expansion)*

Pour chaque ronde, une clé de ronde unique *(K_i)* est dérivée de la clé principale de chiffrement *(*128*,* 192 *ou* 256 *bits)* en utilisant un générateur de clé. Une clé de 128 bits est générée pour chaque ronde, plus une clé supplémentaire pour l'initialisation.

Ronde initiale

- **AddRoundKey** : Chaque bits du bloc de données en cours de chiffrement est `XOR` avec la clé de la ronde (K_i)

Rondes principales (9, 11 ou 13 rondes selon la taille de la clé)

Chaque ronde principale utilise les étapes suivantes :

- **SubBytes** : **Substitution non linéaire** où chaque octet de l'état est remplacé par un autre octet selon une table de correspondance *(table de substitution ou **S-box**)*. Cette substitution renforce la diffusion des données.

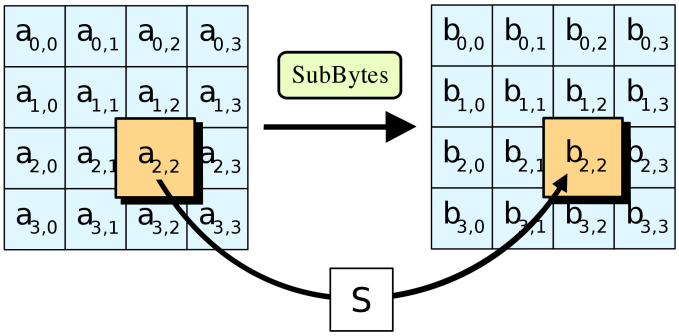


Figure 2: Opération SubBytes [7]

- **ShiftRows** : Effectue une transposition cyclique des lignes de l'état. La première ligne reste inchangée, la deuxième est décalée d'un octet vers la gauche, la troisième de deux octets, et la quatrième de trois octets. Cela contribue à la confusion en modifiant l'ordre des données.

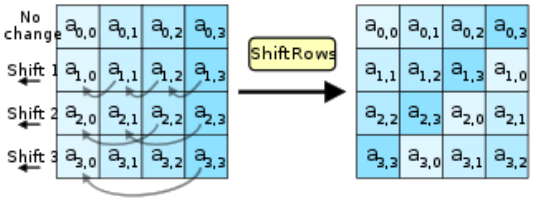


Figure 3: Operation ShiftRows [7]

- **MixColumns** : Opération de mélange linéaire sur les colonnes du bloc. Chaque colonne est multipliée par la matrice hexadécimal 4×4 fixe suivante : $C = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix}$

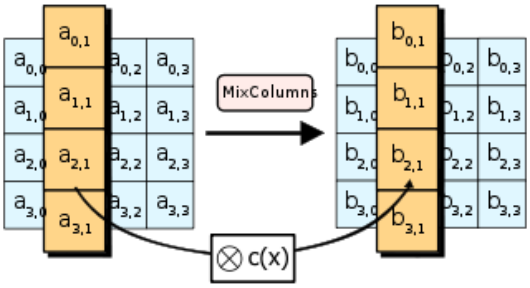


Figure 4: Operation MixColumns [7]

- **AddRoundKey** : Comme dans la ronde initiale, chaque octet de l'état est combiné avec l'octet correspondant de la clé de ronde.

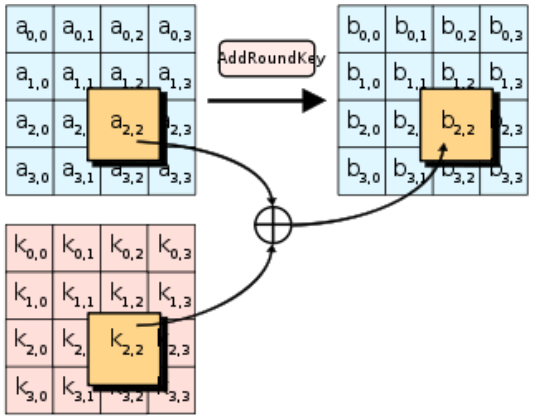


Figure 5: Operation AddRoundKey [7]

Dernière ronde

La dernière ronde diffère des autres rondes car elle n'inclut pas l'étape *MixColumns*. Elle est composée uniquement des opérations :

- **SubBytes**
- **ShiftRows**
- **AddRoundKey**

Vecteur d'initialisation

Un vecteur d'initialisation (IV) est utilisé dans la majorité des systèmes de chiffrement par blocs. C'est le cas pour AES. Son utilisation permet d'obtenir deux chiffrés (C) différents à partir du même texte clair (P) à condition de ne jamais réutiliser le même IV avec une clé identique. À noter que l'utilisation d'un IV sur un $P > 128$, nécessitera un seul vecteur d'initialisation pour l'ensemble des blocs.

Chaînage

AES fonctionnant avec des blocs de taille 128 bits, il est nécessaire de chaîner les blocs entre eux afin de garantir l'intégrité du message. Il existe plusieurs modes de chaînage (`CBC`, `CFB`, `OFB`), `CBC` étant le plus connu.

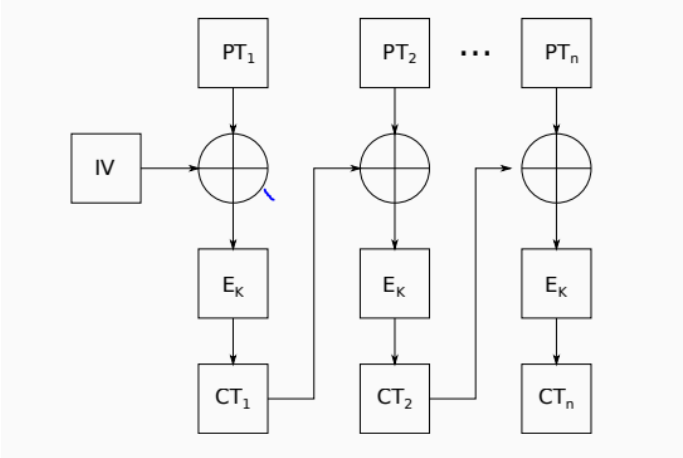


Figure 6: Chaînage CBC [8]

S-box

La Rijndael S-box (S-box) est une méthode permettant d'ajouter de la non-linéarité dans le processus de chiffrement de AES utilisée lors de l'opération `SubBytes()`. S-box est une table de substitution de taille 16×16 permettant de transformer n'importe quelle valeur codée sur 8 bits en une autre valeur.

Utilisation de la S-Box

La première étape consiste à obtenir l'**inverse multiplicatif** de nos bits d'entrées en tant qu'élément du Champ Galois d'ordre 2^8 (GF(2^8)) [9]. Cette étape empêche que pour toute entrée à part 0, on ne puisse obtenir une sortie égale à l'entrée. Après avoir réalisé l'inversion, la deuxième étape est la **transformation affine**. Cette transformation est appliquée pour chaque octet sortant de notre S-box. La transformation affine est définie de la façon suivante:

$$b_i = b_i \oplus b_{(i+4)\ mod(8)} \oplus b_{(i+5)\ mod(8)} \oplus b_{(i+6)\ mod(8)} \oplus b_{(i+7)\ mod(8)} \oplus C_i$$

Avec $0 < i < 8$ où b_i est le $i^{ème}$ bit de l'octet et c_i le $i^{ème}$ bit de l'octet c, une constante possédant la valeur 63_{10}.

1.2 Le chiffrement RSA

Rivest–Shamir–Adleman (RSA) est un crypto-système à clé publique décrit pour la première fois en 1977 par trois chercheurs : Ron **Rivest**, Adi **Shamir** et Leonard **Adleman**. Ce crypto-système utilise une paire de **clé publique** et **clé privée** pour le chiffrement et le déchiffrement des données. Il s'appuie sur la résolution du **problème difficile de factorisation des grands entiers** pour assurer sa sécurité. RSA peut être aussi utilisé pour la création et la vérification de signatures électroniques. Contrairement aux messages, les signatures sont créées à l'aide de la clé privée et vérifiées à l'aide de la clé publique, tandis que les messages sont chiffrés à l'aide d'une clé publique et déchiffrés à l'aide d'une clé privée.

Bob	*Alice*
Création de la clé	
Choisir et garder secret deux ***nombres premiers*** p *et* q. *Choisir un* ***exposant de chiffrement*** e *tel que* $\gcd(e,(p-1)(q-1))=1$. *Publier* $N=pq$ *et* e.	
Chiffrement	
	Choisir un ***texte clair*** m. *Utiliser la* ***clé publique*** (N,e) *pour calculer* $c \equiv m^e \pmod N$. *Envoyer* le **chiffré** c à Bob.
Déchiffrement	
Calculer d *tel que* $ed \equiv 1 \pmod{(p-1)(q-1)}$. *Calculer* $m' \equiv c^d \pmod N$. Ainsi $m'=m$	

Table 1: Chiffrement RSA [2]

2 Attaques par analyse de consommation électrique

Definition 2.1 (Attaques par analyse de consommation électrique) *Les attaques par analyse de consommation électrique exploitent le fait que la consommation électrique instantanée d'un dispositif cryptographique dépend des* ***données*** *qu'il traite et des* ***opérations*** *qu'il effectue [3].*

Paul Kotcher figure parmi les pionniers des attaques par canaux auxiliaires, ayant contribué à la rédaction de plusieurs articles scientifiques influents dans ce domaine [10, 11, 12].

Lors de la conférence CRYPTO '99, Paul Kotcher et al. [11] ont introduit deux concepts majeurs dans leur article : **Simple power analysis (SPA)** et **Differential power analysis (DPA)**.

Suite à cette publication, la recherche dans le domaine des attaques par analyse de consommation électrique s'est accélérée et c'est lors du CHES 2004 que la troisième méthode majeure utilisée dans ce domaine a été introduite: **Correlation power analysis (CPA)** [13].

Nous pouvons distinguer deux catégories de SCA, les **attaques par canaux auxiliaires avec profilage** et les **attaques par canaux auxiliaires sans profilage**.

Definition 2.2 (Attaques par canaux auxiliaires avec profilage) *Une attaque par profilage consiste en deux étapes. Premièrement, l'adversaire se procure une copie du matériel cible et définit les fuites physiques. Deuxièmement, il réalise l'attaque sur la cible, afin de retrouver la clé [14].*

Definition 2.3 (Attaques par canaux auxiliaires sans profilage) *L'attaquant a uniquement accès aux fuites physiques capturées sur la machine cible. Pour retrouver la clé secrète utilisée, il va utiliser l'analyse statistique pour détecter les dépendances entre les fuites mesurées et les données sensibles [14].*

Avec l'évolution rapide de l'Intelligence Artificielle (IA) dans la dernière décennie, les attaques par canaux auxiliaires n'ont pas échappé à son utilisation. Dans un premier temps avec l'utilisation du Ma-

chine Learning (ML) [15, 16, 17] en utilisant principalement deux algorithmes d'apprentissage: Support Vector Machine (SVM) et Random Forest (RF).

Suivant le succès du ML pour attaquer des traces très bruyantes, le Deep Learning (DL) a fait son apparition avec l'utilisation majoritaire des Convolutional Neural Network (CNN).

2.1 Simple power analysis (SPA)

Definition 2.4 (Simple power analysis) *Technique qui consiste à interpréter directement les mesures de consommation d'énergie recueillies au cours d'opérations cryptographiques [11].*

Les algorithmes de chiffrement sont majoritairement exécutés de manière séquentielle. Ces algorithmes sont implémentés en utilisant une suite d'instructions unique, qui est ensuite transformée en instructions machine. Cette suite d'instructions machines est ensuite interprétée et exécutée par le micro-contrôleur. Si l'on mesure la consommation électrique de ce micro-contrôleur à l'exécution de l'algorithme, nous verrons des motifs propres à la consommation des instructions de l'algorithme cryptographique en cours d'exécution. Cette mesure complète s'appelle une **trace** et peut être un vecteur d'attaque potentiel sur des systèmes embarqués.

Les attaques SPA peuvent être de différents types:

- Inspection visuelle de la consommation électrique (2.1.1);
- Attaques SPA basées sur des modèles (2.1.2);
- Attaques par collisions (2.1.3).

2.1.1 Inspection visuelle de la consommation électrique

L'inspection visuelle de traces de consommation électrique permet, si la trace n'est pas trop bruitée, de révéler des informations sur un ou plusieurs secrets d'un algorithme cryptographique[1]. Il est possible de dériver visuellement la clé d'un système cryptographique si les opérations de l'algorithme dépendent de celle-ci[2].

[1] La révélation du secret peut être partiel en fonction de la qualité de la trace.

[2] Dans notre cas, de la consommation électrique de notre système lors de l'exécution de l'algorithme cryptographique.

Exemple pratique

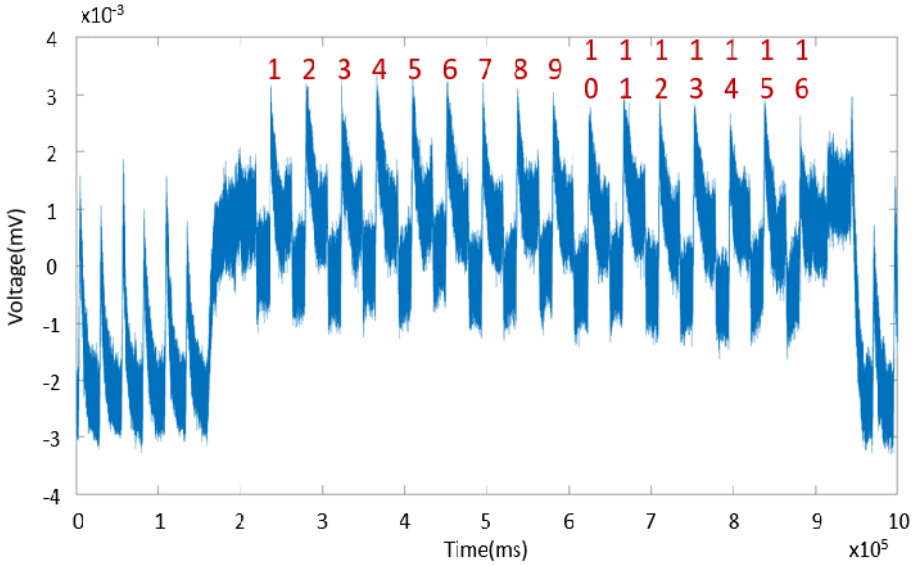


Figure 7: SPA permettant de visualiser les 16 rondes de DES [18]

Cet extrait de trace permet clairement de visualiser les 16 rondes de DES.

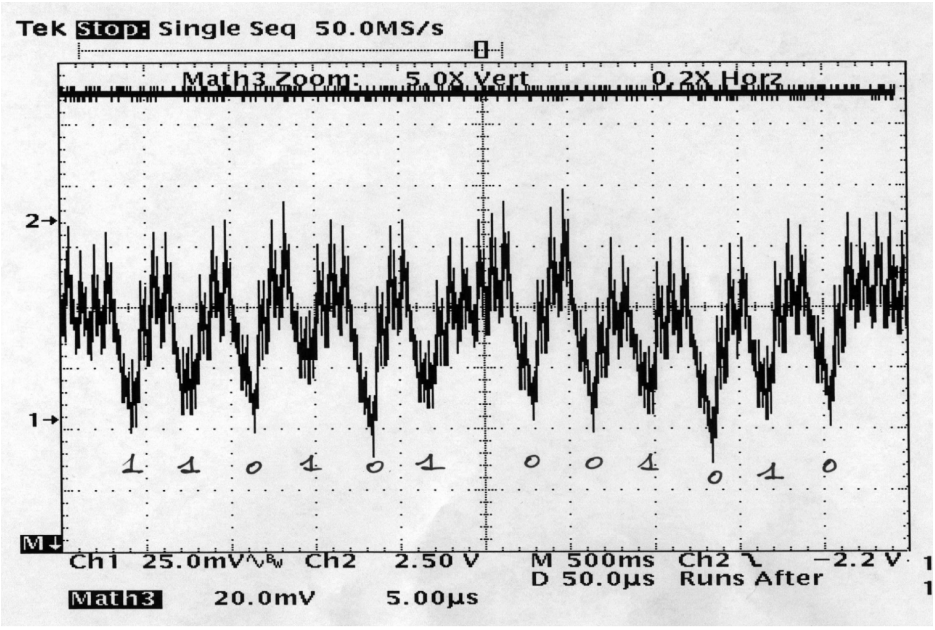


Figure 8: Visualisation d'une partie de la clé de chiffrement d'un Triple-DES grâce à l'attaque SPA [19]

Si nous regardons ce qui se passe en zoomant sur une des rondes, on aperçoit différentes variations, ces variations constantes et distinctes nous permettent d'identifier des bits et donc de pouvoir inférer de l'information sur la clé secrète.

Pour la figure 8, on remarque que si le bit de la clé est 0, il y a un pic simple dans le bas de la trace, au niveau du 1 de l'ordonnée.

2.1.2 Attaques SPA basées sur des modèles *(Template-based SPA attacks)*

Definition 2.5 (Template Attacks) *Les templates attacks exploitent le fait que la consommation énergétique dépend des données qui sont traitées. Dans une attaque basée sur des modèles, nous caractériserons les traces de puissance par une loi normale multidimensionnelle [3].*

Ce type d'attaque est un sous-ensemble des attaques par profilage. Elles sont très puissantes, car nous supposons que nous avons préalablement un accès total à une copie du système cible pour régler avec précision tous les paramètres de l'attaque.

Une fois que nous avons le système cible, nous pouvons créer un modèle *(template)* en récupérant un grand nombre de traces de consommation d'énergie[3]. C'est ce que l'on appelle la phase de prétraitement *(pre-processing)*.

Une fois la phase de prétraitement terminée, nous n'aurons besoin que d'un très petit nombre de traces pour réaliser l'attaque.

Voici les quatre étapes pour réaliser une Template Attacks[20]:

1. Récupérer un grand nombre de traces de consommation d'énergie en utilisant différentes entrées *(textes clairs, clés)*[4];
2. Créer un **modèle** du système cible. Le modèle va indiquer quelques "points d'intérêt"[5] dans les traces de puissance et va utiliser la loi normale multidimensionnelle sur chaque point des traces de puissances;
3. Sur le **système cible**, nous allons enregistrer un petit nombre de traces de consommation énergétique en utilisant différents textes clairs[6];
4. Utiliser notre modèle sur les traces récoltées pendant l'attaque. Pour chaque sous-clé, chercher quelle valeur est la plus probable d'être la bonne sous-clé. Continuer jusqu'à ce que la totalité de la clé soit retrouvée;

Definition 2.6 (Loi Normale) *On dit qu'une variable aléatoire X suit la loi normale[21] de paramètres $m \in \mathbb{R}$ et σ^2, avec $\sigma > 0$, ce que l'on note $X \hookrightarrow N(m, \sigma^2)$, si elle est continue et admet pour densité :*

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

Definition 2.7 (Loi Normale Multidimensionnelle) *Généralisation multidimensionnelle de la loi normale [22].*

$$f_{\mu,\sigma}(x) = \frac{1}{(2\pi)^{N/2}\det(\Sigma)^{1/2}} \exp\left[-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right]$$

Avec vecteur $\mu \in \mathbb{R}^N$ représentant son centre, $\Sigma \in M_N(\mathbb{R})$ une matrice semi-définie positive qui est sa matrice de variance-covariance et un vecteur $x \in \mathbb{R}^N$.

[3] Cela peut nécessiter des dizaines de milliers de traces de puissance.

[4] Il faut s'assurer que nous avons enregistré suffisamment de traces pour avoir des informations sur chaque valeur de sous-clé.

[5] Points qui contiennent le plus d'informations sur l'instruction exécutée.

[6] Cette fois nous n'avons pas accès à la clé, elle est fixée préalablement.

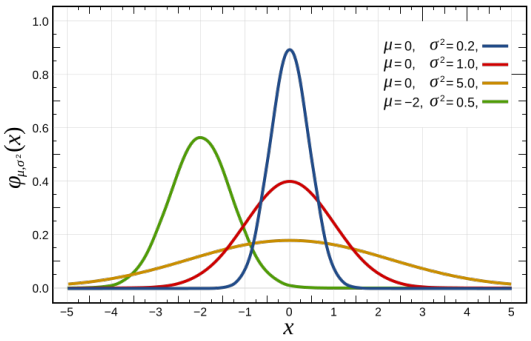


Figure 9: Différentes densités de lois normales en deux dimension [22]

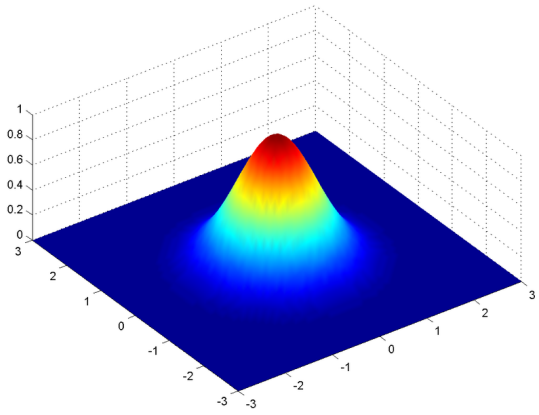


Figure 10: Densité d'une loi gaussienne en 3 dimension [22]

2.1.3 Attaques par collisions

Supposons que nous ayons la possibilité d'observer la consommation énergétique pour le chiffrement de deux textes clairs p_i et p_i^*, avec la clé k_j qui est inconnue. Une attaque par collisions consiste en l'utilisation du fait que lorsque deux textes clairs sont chiffrés avec la même clé, il existe potentiellement une valeur intermédiaire $v_{i,j}$ égale à: $v_{i,j} = f(p_i, k_j) = f(p_i^*, k_j)$.

Si une telle valeur intermédiaire existe, on dit que la **valeur intermédiaire entre en collision**.

Information

Il est important de mentionner que pour deux textes clairs p_i et p_i^*, une collision ne peut se produire pour toutes les valeurs de la clé, mais seulement pour un certain sous-ensemble de clés.

Ainsi, chaque collision permet de réduire l'espace de recherche pour la clé. Si plusieurs collisions sont observées, la clé peut être ainsi retrouvée.

En pratique si nous voulons détecter une collision, nous devons dans un premier temps identifier la partie de la trace de consommation électrique dans laquelle la valeur intermédiaire entrant en collision est calculée.

Suite à cela, nous allons comparer deux traces de puissance et décider si ces deux parties sont égales ou non. Pour faire cela, nous pouvons utiliser l'une des deux méthodes suivantes:

1. Supposons que nous ayons accès préalablement au système cible avant *(Template Attacks)*. Nous allons créer un modèle pour la partie de la trace dans laquelle il y a une valeur intermédiaire entrant en collision. Si nous travaillons avec des octets, nous devons construire 256 modèles. Lors de l'attaque, nous allons essayer d'associer les modèles avec les deux traces observées. **Une collision a lieu si un même modèle correspond le mieux aux deux traces**;

2. Supposons que nous ayons accès uniquement aux deux traces de consommation énergétique, et que nous savons dans quelle partie de la trace la valeur intermédiaire entrant en collision est traitée. Nous pouvons donc extraire cette partie *(ou quelques points de cette partie)* dans les deux traces *(modèle réduit)* et les associer entre elles. Pour trouver la collision, il ne nous reste plus qu'à utiliser le LSQ[7] Test.

[7] Méthode des moindres carrés en français.

Definition 2.8 (Least Square *(LSQ)* Test) *Méthode qui consiste à trouver les paramètres minimisant*

$$\sum_{t=t_0}^{t_0+N-1} (p[t,0] - p[t;\delta])^2$$

où $p[x,y]$ *est un octet du texte clair,* t *la trace de consommation,* N *le nombre de points d'échantillonnage et* $\delta = 1 \dots 255$ *[23, 24].*

2.1.4 SPA pour RSA

Le chiffrement RSA nécessite d'élever des nombres à certaines puissances modulo un entier. Cette opération est utilisée pour le chiffrement et pour le déchiffrement de chaque bloc de message, il est donc important de pouvoir la réaliser aussi rapidement que possible [25].

Un algorithme très connu pour calculer la puissance d'un nombre est le *square-and-multiply*[8].

```
#include <string>
#include <cinttypes>

using namespace std;

/**
* Square-and-multiply algorithm
*
* Efficient method for calculating the power of a number
*
* @param x: Base number
* @param d: Binary expansion of exponent d
* @return y = x^d
*/
uint64_t squareAndMultiply(uint64_t x, string d) {
  uint64_t r0 = 1;
  uint64_t r1 = x;
  int i = 0;

  while (i <= (d.length() - 1)) {
    r0 = r0 * r0;
    if (d.at(i) == '1')
      r0 = r0 * r1;

    i++;
  }

  return r0;
}
```

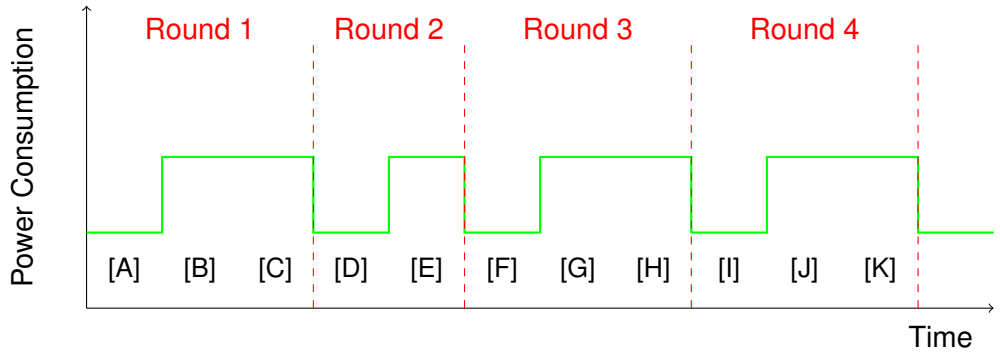
Listing 1: Algorithme square-and-multiply

Cependant, si l'on étudie cet algorithme dans le but de réaliser une attaque par canaux auxiliaires, nous remarquons rapidement que le nombre d'opérations pour réaliser l'exponentiation diffère en fonction du nombre de 0 ou de 1 dans la représentation binaire de notre exposant.

[8] Il suffit de réaliser les opérations modulo pour faire de l'exponentiation modulaire rapide.

Si le bit en cours de traitement est un 1, nous réalisons une opération supplémentaire *(multiply)*.

Cette opération supplémentaire pouvant être visualisée grâce à un oscilloscope, nous pouvons ainsi retrouver les bits de l'exposant utilisé pour le chiffrement RSA.



Dans cette simulation de l'observation de 4 itérations de la fonction square-and-multiply [26] utilisée dans certaines implémentations de RSA, nous pouvons clairement distinguer les itérations où il y a les opérations `square` et `multiply` *(Round 1, 3 et 4)*, aux itérations où il n'y a que l'opération `square` *(Round 2)*.

Ainsi, une attaque SPA permet de découvrir l'exposant (d) utilisé dans une opération de déchiffrement RSA, ce qui permet de casser le chiffrement.

2.2 Differential power analysis (DPA)

Definition 2.9 (Differential power analysis) *Technique exploitant la dépendance entre la consommation d'énergie des dispositifs cryptographiques et les données. Elle utilise un grand nombre de traces de consommation d'énergie pour analyser celle-ci à un moment donné en fonction des données traitées. [3].*

Les attaques DPA sont très populaires parmi les attaques par analyse de consommation électrique, car elles ne nécessitent pas de connaissances détaillées sur le système attaqué. Il suffit en général de connaître l'algorithme cryptographique exécuté sur le système.

SPA	DPA
Une seule ou un petit nombre de traces de consommation.	Un grand nombre de traces de consommation.
Ne fonctionne pas sur des traces avec du bruit.	Fonctionne bien sur des traces avec beaucoup de bruit.
Nécessite une connaissance détaillée du système cryptographique attaqué.	Dans la majorité des cas, ne nécessite pas une connaissance détaillée sur le système cryptographique attaqué.
La consommation électrique d'un système est principalement analysée sur l'axe du temps.	Se concentre uniquement sur la dépendance des données avec la trace de consommation électrique.
Il n'y a pas de stratégie d'attaque générale qui est utilisée par toutes les attaques SPA.	Il y a une stratégie d'attaque principale utilisée par toutes les attaques DPA.

Table 2: Différences entre les attaques SPA et DPA

Nous allons maintenant décrire les différentes étapes nécessaires pour réaliser une Differential power analysis (DPA) [3]:

1. Choisir un résultat intermédiaire de l'algorithme exécuté;

2. Mesurer la consommation électrique;
3. Calculer les valeurs intermédiaires hypothétiques;
4. Faire correspondre les valeurs intermédiaire avec les valeurs de consommation électrique;
5. Comparer les valeurs hypothétiques de consommation électrique avec les traces de consommation électrique.

Ci-dessus, nous avons la stratégie d'attaque principale utilisée par toutes les attaques DPA ; dans la suite, nous allons étudier en détail chacune de ces étapes.

Étape 1: Choisir un résultat intermédiaire de l'algorithme exécuté

Le résultat intermédiaire choisi est une fonction $f(d, k)$, où d est une **donnée non-constante**[9] et k est une **petite partie de la clé**.

Étape 2: Mesurer la consommation électrique

La mesure de la consommation électrique se réalise pendant le chiffrement ou le déchiffrement de D blocs de données différents.

Pour chaque opération de chiffrement ou de déchiffrement, l'attaquant doit connaître la valeur de la donnée d qui est impliquée dans le calcul du résultat intermédiaire *(2.2)*. On écrit les valeurs des donnés connues dans une liste $d = (d_i, \ldots, d_D)'$, telle que d_i correspond à la valeur de la donnée de la $i^{\text{ème}}$ exécution de la fonction de chiffrement ou déchiffrement.

Lors de chaque exécution, l'attaquant sauvegarde la trace correspondante au bloc de données d_i dans une liste $t'_i = (t_{i,1}, \ldots, t_{i,T})$, où T correspond à la taille de la trace.
Nous pouvons écrire une matrice **T** de taille $D \times T$.

Information

Pour réaliser une attaque DPA, les traces doivent être alignées, c'est-à-dire que la valeur de la consommation électrique t_j de la matrice **T** doit correspondre à la même opération.

Étape 3: Calculer les valeurs intermédiaires hypothétiques

Nous notons les valeurs intermédiaires hypothétiques $k = (k_1, \ldots, k_K)$ où K représente le nombre total de choix possibles pour k. Les composants du vecteur k peuvent être vus comme les différentes hypothèses de clé.

Le vecteur de donnée d et les hypothèses de clé k peuvent être utilisés pour calculer facilement les valeurs intermédiaires hypothétiques $f(d, k)$ pour toutes les opérations de chiffrement D et pour toutes les clés d'hypothèses K.
Le résultat peut être stocké dans une matrice **V** de taille $D \times K$.

[9] Dans la plupart des cas, c'est un *texte clair*, ou *chiffré*

Étape 4: Faire correspondre les valeurs intermédiaires avec les valeurs de consommation électrique

Dans cette étape, nous allons mettre en correspondance les valeurs intermédiaires hypothétiques **V** avec la matrice **H** correspondant aux valeurs hypothétiques de consommation d'énergie.

Pour ce faire, l'attaquant peut utiliser plusieurs techniques de simulation comme la Distance de Hamming ou les Poids de Hamming.

En utilisant l'une de ces techniques, la consommation d'énergie de l'appareil pour chaque donnée intermédiaire hypothétique $v_{i,j}$ est simulée afin d'obtenir une valeur hypothétique de consommation d'énergie $h_{i,j}$.

Information

La qualité de la simulation dépend fortement des connaissances de l'attaquant concernant l'appareil analysé. Plus la simulation est proche des caractéristiques de la consommation d'énergie réelle de l'appareil, plus l'attaque par DPA sera efficace.

Étape 5: Comparer les valeurs hypothétiques de consommation électrique avec les traces de consommation électriques

Après avoir fait correspondre **V** à **H** nous pouvons comparer toutes les colonnes $h_i \in H$ avec les colonnes $t_j \in T$.

Le résultat de cette comparaison est stocké dans la matrice **R** de taille $K \times T$, où chaque élément $r_{i,j}$ contient le résultat de la comparaison entre la colonne h_i correspondant à la *valeur hypothétique de la consommation d'énergie pour chaque clé d'hypothèse* et t_j correspondant aux *traces à chaque position*.

Information

Plus la valeur de $r_{i,j}$ est élevée, plus les colonnes h_i et t_j sont similaires.

Sachant que les traces de consommation d'énergie correspondent à la consommation électrique du système lorsqu'il exécute un algorithme cryptographique en utilisant différentes données en entrée. Et que les résultats intermédiaires choisis lors de l'étape 1 *(2.2)* font partie de cet algorithme.

On définit ck l'index de la clé utilisée et ct le moment où la trace correspond à l'utilisation de la clé.

On sait, à partir des étapes précédentes, que la consommation d'énergie hypothétique h_{ck} a été simulée par l'attaquant à partir de la valeur de v_{ck}. Ainsi, les colonne h_{ck} et t_{ct} sont fortement reliées et donnent la plus grande valeur de la matrice **R** $r_{ck,ct}$. Toutes les autres valeurs de **R** sont basses, car les autres colonnes de **H** et **T** ne sont pas fortement reliées.

L'attaquant peut révéler la clé utilisée sur le système en regardant les plus grands indices dans la matrice **R**.

Information

En pratique, il est possible que toutes les valeurs de la matrice **R** soient approximativement les mêmes. Cela signifie que l'attaquant n'a pas mesuré suffisamment de traces de consommation d'énergie pour faire une estimation correcte de la relation entre les colonnes de **H** et **T**.

2.2.1 DPA pour RSA

Il existe différentes attaques DPA visant RSA. La principale méthode vise l'implémentation du calcul de l'exposant d. Cette attaque fonctionne sur plusieurs implémentations de RSA, mais nous allons nous concentrer sur le plus basique, à savoir le calcul de l'exposant à l'aide de square-and-multiply dont nous avons vu le fonctionnement plus tôt dans l'explication de l'attaque SPA sur RSA. C'est cet algorithme qui va nous fournir un vecteur d'attaque sur RSA, car les opérations s'effectuent bit à bit, ce qui signifie que nous pouvons tenter de recouvrir des informations.

Hypothese initiale

La première étape dans le but de retrouver d est d'effectuer une hypothèse sur son premier bit.

Nous avons deux possibilités:

- $b_0 = 0$ On mets au carré
- $b_0 = 1$ On mets au carré et on multiplie

Avant de pouvoir comparer nos hypothèses et nos mesures il faut expliquer la **fonction de sélection**.

Fonction de sélection

La fonction de sélection, notée $d(x, j)$, est calculée sur les résultats intermédiaires (par exemple, un reste ou un résultat de carré/multiplication) pour chaque hypothèse j.

Elle dépend :

- Des données d'entrée x.
- De la structure hypothétique de la clé privée.

On peux utiliser le poids de Hamming $W(x)$ (nombre de bits égaux à 1) car il est corrélé à la consommation d'énergie d'un système. La fonction s'évalue de la manière suivante :

$$d(x) = \begin{cases} -1 & \text{si } W(x) < E(n), \\ 0 & \text{si } W(x) = E(n), \\ +1 & \text{si } W(x) > E(n). \end{cases}$$

- $E(n)$ représente la valeur moyenne du poids de hamming espéré, $E(n) = n/2$ pour une architecture n- bits

La fonction classe les résultat en trois classes, valeur faible (-1), valeur neutre (0), valeur forte (+1).

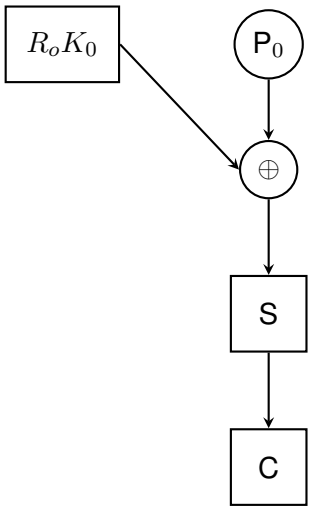
Corrélation [27]

L'application de la fonction de sélection nous permets maintenant de corréler nos valeurs en sortie $d(x, j)$ avec les traces capturés $P(x, t)$. Pour se faire, nous pouvons utiliser l'indice de corrélation de Pearson.

2.2.2 DPA pour AES [1]

Introduction

Pour réaliser cette attaque, seulement deux étapes de l'algorithme AES vont nous intéresser : **AddRoundKey** et **SubBytes**. Nous partons du principe que nous connaissons le texte clair P et le texte chiffré C.



Information

- R_o et K_0 sont les entrées de **AddRoundKey** ($\oplus$).
- P_0 représente un fragment du texte clair.
- S représente l'opération **SubBytes**.

Nous partons du principe que P_0 est connu et R_0K_0 inconnu et qu'en arrivant à trouver un bit de C, on est en capacité de trouver R_0K_0. Cette attaque fonctionne dans l'implémentation la plus basique de AES, dans laquelle les données sont traitées octets par octets.

Etape 1: Test exhaustif de la clé

Nous partons du principe que nous possédons plusieurs traces mesurant le chiffrement de différents textes clairs *(P)* avec la même clé secrète *(K)*. Si nous regardons nos différentes traces de consommation électrique, il est impossible d'inférer de quelconques informations à cause du bruit. Afin de réduire le bruit, nous allons pour chaque octet du bloc de données de P, essayer de deviner une partie de la clé en testant exhaustivement les clés sur ce bloc de données. On commence par le premier octet. P_0 avec notre clé R_0K_0 que l'on essaie de deviner. Par exemple,

- $P_0 = 0xC7$
- Hypothèse, $K_0 = 0x00$
- $SBox(P_0 \oplus K_0) = 0xC6$
- $MSB(0xC6) = 1$

On classe ensuite le résultat en fonction d'un signe distinctif, par exemple la valeur de son Most Significant Bit (MSB).

Etape 2: Analyse des traces

t Une fois que nous avons constitué nos groupes pour un octet, nous allons utiliser ces traces pour créer une **trace de consommation moyenne** pour chaque classe. Si on soustrait nos deux traces moyennées, nous allons obtenir une **trace finale** où est enlevé le bruit causé par le système. Il ne nous reste plus qu'à analyser cette trace de consommation finale pour essayer de repérer des pics de consommation distinctifs, afin de déduire un octet de la clé.

L'objectif est ensuite d'essayer de deviner les 16 octets de la clé secrète. L'avantage de cette stratégie est de pouvoir réduire l'espace de clés de 2^{128} pour une implémentation $AES{-}128$ à une recherche exhaustive de $16 \cdot 256 = 4096$ possibilités.

2.3 Correlation Power Analysis (CPA)

L'attaque par CPA a été introduite lors de la conférence CHES 2004 par Eric Brier, Christophe Clavier, et Francis Olivier [13].

Cette attaque peut être décrite en 4 étapes, afin de retrouver la clé utilisée [28]:

1. Créer un modèle correspondant à la consommation électrique du système cible.
   1.1 Le modèle va examiner un point spécifique de l'algorithme de chiffrement.
2. Demander au système cible de chiffrer plusieurs textes clairs différents. Enregistrer une trace de la consommation d'énergie du système cible pendant chacun des chiffrements.
3. Attaquer les sous-clés de la clé secrète.
   3.1 Considérer toutes les options possibles pour la sous-clé. Pour chaque supposition et chaque trace, utiliser le texte en clair connu et la sous-clé supposée pour calculer la consommation d'énergie selon notre modèle.
   3.2 Calculer le coefficient de corrélation de Pearson entre la consommation énergétique modélisée et la consommation réelle. Effectuer cette opération pour chaque point de données dans les traces.
   3.3 Décider quelle est la sous-clé qui correspond le mieux aux traces mesurées.
4. Rassembler toutes les sous-clés sélectionnées dans l'étape précédente pour récupérer la totalité de la clé secrète.

Definition 2.10 (coefficient de corrélation de Pearson) *Le coefficient de Pearson est un indice reflétant une relation linéaire entre deux variables continues [13, 28, 29].*

$$\rho_{WH} = \frac{cov(W,H)}{\sigma_w \sigma_H} = \frac{E[(W-\mu_W)(H-\mu_H)]}{\sqrt{E[(W-\mu_W)^2]E[(H-\mu_H)^2]}}$$

Les attaques par DPA et CPA utilisent toutes les deux la relation entre la consommation énergétique et les données. Cependant les attaques par DPA vont **viser la différence des moyennes des courbes**, tandis que les attaques par CPA vont **chercher la meilleure corrélation pour toutes les valeurs de clef** [30].

2.3.1 CPA pour AES-128

Comme expliqué lors de l'introduction *(1.1)*, AES est composé de plusieurs rondes, suivant un motif spécifique:

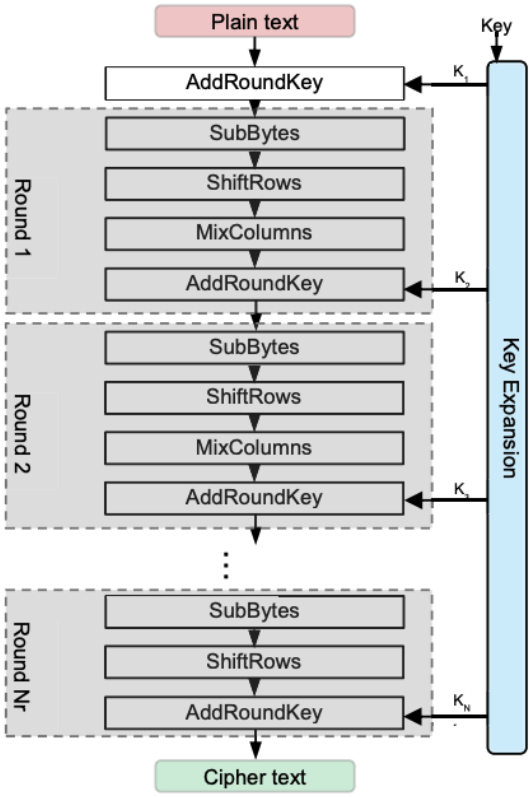


Figure 11: Algorithme AES [31]

Une attaque par CPA sur `AES-128` peut être réalisée en observant la consommation électrique uniquement après le premier `SubBytes` [28].

Définissons un état s, initialisé à la valeur du texte clair avant que le chiffrement ne soit exécuté $(s = plaintext)$.

Après la première fonction `AddRoundKey` et `SubBytes`, notre état est égal à : $s = sbox[plaintext \oplus key]$[10].

Nous savons que la `sbox` consiste à regarder la valeur d'un élément dans un tableau qui est connue de tous. C'est donc un point d'attaque efficace.

[10] Avec $i \in [0, 256]$.

Lors de la création de notre modèle, nous connaissons l'octet du texte clair *(p_d)*. Ainsi, nous pouvons modéliser notre consommation d'énergie de la manière suivante:

$$h_{d,i} = Poids\_de\_Hamming(sbox[p_d \oplus i])$$

.

En créant ce modèle, nous attaquons un octet de la clé à la fois. Nous devons donc tester $2^8 = 256$ valeurs différentes pour chaque sous-clé. Dans `AES-128`, la clé est composée de 16 octets, donc pour retrouver l'intégralité de la clé, nous devrons tester au maximum $16 \times 2^8 = 2^{12} = 4096$ valeurs.

Conclusion

CPA nous offre une amélioration considérable pour `AES-128`, comparé à une attaque par force brute qui nous ferait tester 2^{128} valeurs au maximum pour retrouver la clé.

2.4 L'utilisation du machine learning et deep learning dans les attaques par canaux auxiliaires

Comme évoqué dans l'introduction de ce chapitre, l'IA a fait son apparition dans les SCA et elle est devenue une technique précieuse pour les attaques par canaux auxiliaires avec profilage. L'utilisation du ML et du DL a permis la mise en place d'attaques plus performantes, pouvant être utilisées sur des mesures à très hautes dimensions et qui sont résistantes à la déformation du signal comme la gigue *(pour les CNN)* [14].

De nombreux papiers ont été publiés afin de présenter un nouveau modèle qui performait mieux que les précédents pour certaines traces[11]. Cependant, l'un des papiers les plus influents dans le domaine des attaques par canaux auxiliaires utilisant des méthodes de Deep Learning est *"Study of Deep Learning Techniques for Side-Channel Analysis and Introduction to ASCAD Database"* [14]. En proposant une base de données pouvant être utilisée comme benchmark pour comparer l'ensemble des modèles, ainsi que certains modèles de base pouvant servir de base pour le développement de modèles plus performants, Ryad Benadjila et al. ont permis une accélération des recherches dans ce domaine.

Il ne faut également pas omettre la conférence CHES qui chaque année depuis 2015 propose un challenge de crypto-ingénierie qui amène à la création de nouveaux modèles, afin de casser l'implémentation de l'algorithme de chiffrement en fonction des caractéristiques du matériel utilisé pour créer le challenge.

Dans la première partie de ce projet, nous réaliserons une revue de la littérature sur le sujet choisi. Cette étape vise à définir les concepts fondamentaux et à établir les bases nécessaires à l'implémentation, qui se concentrera dans un second temps sur les attaques et les défenses liées aux canaux auxiliaires, notamment via l'analyse de consommation électrique pour les chiffrements AES et RSA.

Pour ce rapport, nous nous sommes limités aux bases théoriques et méthodologiques pour attaquer différentes implémentations. Une analyse plus détaillée des bases de données et des modèles sera réservée au second rapport, afin d'éviter de surcharger ce document avec des informations qui seraient davantage pertinentes pour le second semestre.

[11] Les modèles peuvent être plus performants pour un système cryptographique spécifique ou pour des traces plus bruyantes par exemple.

2.5 Réseau neuronal convolutif

Definition 2.11 (réseaux de neurones convolutifs) *Ils utilisent des données tridimensionnelles pour les tâches de* ***classification d'images et de reconnaissance d'objets****. Ils se distinguent des autres réseaux neuronaux par leurs* ***performances supérieures avec des entrées de signaux d'image, de parole ou audio*** *[32].*

Les réseaux de neurones convolutifs se révèlent extrêmement utiles pour examiner les profils de consommation énergétique, car ils excellent dans la classification d'images, ce qui leur permet de détecter des différences subtiles entre les traces et d'en extraire des éléments essentiels.

Un CNN est composé de trois couches différentes:

- Couche de **convolution**;
- Couche de **pooling**;
- Couche **entièrement connectée** *(FC)*;

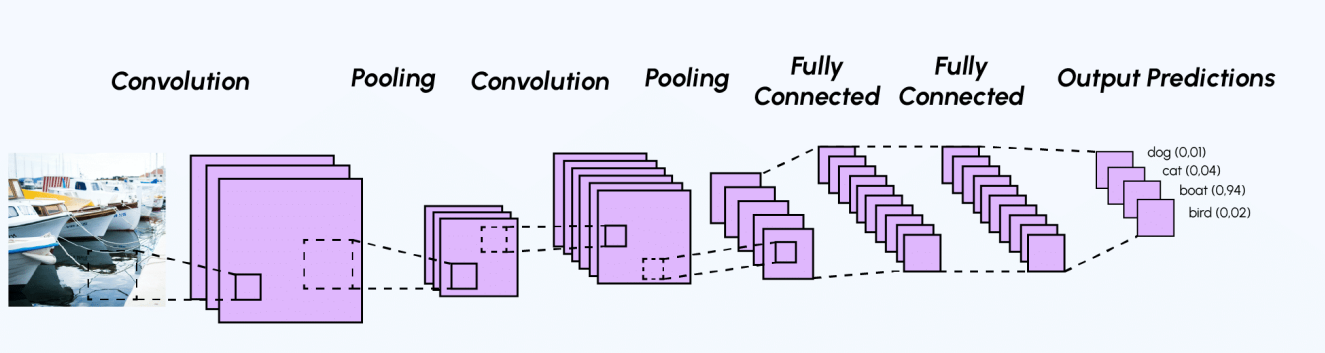


Figure 12: Exemple de fonctionnement d'un CNN [33]

2.5.1 Couche de convolution

Cette couche nécessite des **données d'entrée**, un **filtre** et une **carte de caractéristiques** *(feature map)* pour son bon fonctionnement.

Le filtre va se déplacer sur l'image pour vérifier si la caractéristique recherchée est présente. C'est ce qu'on appelle la **convolution**.

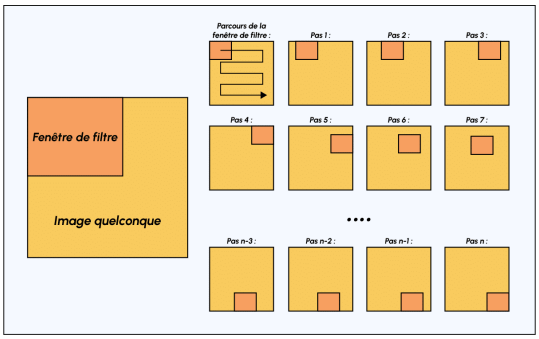


Figure 13: Parcours de la fenêtre de filtre [33]

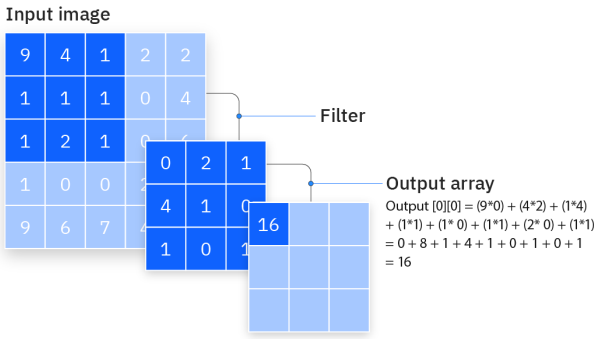


Figure 14: Application d'un filtre sur une image [32]

La taille du filtre peut varier, cependant, dans la majorité des cas, c'est une matrice 3×3. La taille de la matrice utilisée pour filtrer définit également la taille du champ réceptif *(output array)*.

La matrice de sortie est calculée en réalisant le produit scalaire entre les pixels d'entrée et le filtre. Suite à ça, le filtre se décale d'un cran et répète le processus jusqu'à la fin.

Le résultat final est appelé *carte de caractéristiques* ou *feature map*.

Cette couche possède trois hyperparamètres à définir avant le début de l'entraînement du réseau de neurones qui affectent la taille du volume de sortie:

- **Nombre de filtres**: Affecte la profondeur de la sortie. On dira que notre réseau a une profondeur de n, avec n le nombre de filtres.
- **Pas**: Nombre de pixels, que le filtre parcourt sur la matrice d'entrée[12]. Un pas plus grand produit une sortie plus faible.
- **Marge** *(zero-padding)*: Utilisée lorsque les filtres ne correspondent pas à l'image d'entrée. On distingue trois types de marge:
  - **Marge valide**: Abscence de marge, la dernière convolution est supprimée si les dimensions ne s'alignent pas.
  - **Marge identique**: Garantit que la couche de sortie a la même taille que la couche d'entrée.
  - **Marge pleine**: Augmente la taille de la sortie en ajoutant des zéros à la bordure de l'entrée.

2.5.2 Couche de pooling

Cette couche permet de **réduire la dimensionnalité**, ce qui réduit le nombre de paramètres d'entrée.

Encore une fois, un filtre parcourt toute l'entrée. Cependant, cette fois, le filtre n'a aucun poids. Une fonction d'agrégation va être appliquée aux valeurs de la matrice d'entrée pour créer la matrice de sortie. Les deux principaux types de pooling sont:

- **Max-Pooling**[13]: Sélectionne le pixe ayant la valeur maximale.
- **Average-Pooling**: Calcule la valeur moyenne dans le champ réceptif.

[12] La majorité du temps le pas est inférieur à deux.
[13] Cette approche est plus souvent utilisée que l'average pooling.

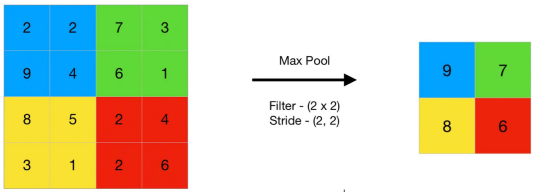


Figure 15: Max-Pooling [33]

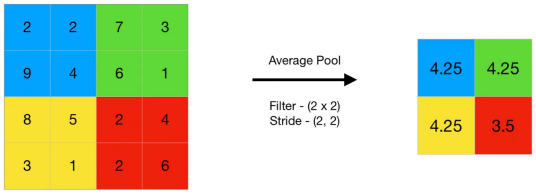


Figure 16: Average-Pooling [33]

Comme vous pouvez le deviner, de nombreuses informations sont perdues dans la couche de pooling. Cependant, cette couche permet de **réduire la complexité**, d'**améliorer l'efficacité** et de **limiter le risque de surajustement** *(overfitting)*.

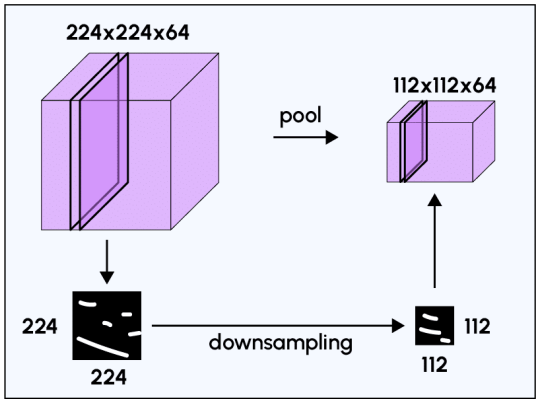


Figure 17: Exemple de l'effet du Max-Pooling [33]

2.5.3 Couche entièrement connectée (FC)

Les autres couches sont partiellement connectées. C'est-à-dire que les valeurs de pixel de l'image d'entrée ne sont pas directement connectées à la couche de sortie.

Pour résoudre cela, il faut une couche entièrement connectée, où chaque nœud de la couche de sortie se connecte directement à un nœud de la couche de sortie.

Ainsi, cette couche est placée à la fin du réseau pour effectuer la classification en fonction des caractéristiques extraites à partir des couches précédentes et de leurs différents filtres.

Alors que les couches de convolution et de pooling ont tendance à utiliser les fonctions ReLu, les couches FC exploitent généralement une fonction d'activation softmax pour classer les entrées de manière appropriée, produisant une probabilité de 0 à 1.

3 Mesures de protection face aux attaques basées sur l'analyse de la consommation électrique

L'objectif des contre-mesures pour les attaques par canaux auxiliaires par analyse de consommation électrique est d'**éviter ou au moins de réduire le lien** entre la **consommation électrique d'un système**

cryptographique et les **valeurs intermédiaires de l'exécution de l'algorithme cryptographique**.

Il existe de nombreuses contre-mesures, et nous allons nous concentrer sur trois d'entre elles:

- Atomicité du canal auxiliaire *(3.1)*;
- Hiding *(3.3)*;
- Masking *(3.4)*.

Pour le Hiding et Masking, nous ferons également une ouverture sur les améliorations qui ont été proposées au fil des ans.

3.1 Atomicité du canal auxiliaire

Comme vu précédemment *(2.1.4)* l'algorithme **Square-and-Multiply** utilisé dans certaines implémentations de RSA est vulnérable à une attaque SPA s'il est implémenté trop simplement.

Pour pallier à ce problème, l'atomicité du canal auxiliaire peut être utilisée afin de ne pas pouvoir distinguer les opérations *square* et *multiply* avec une attaque SPA.

Definition 3.1 (Atomicité du canal auxiliaire) *L'atomicité du canal auxiliaire fait référence à la propriété d'un algorithme ou d'une implémentation cryptographique qui garantit l'exécution d'une opération unique et indivisible (atomicité) sans fuite d'informations sensibles par des canaux involontaires [34].*

3.2 Example d'atomicité du canaux auxiliaire pour RSA

Pour se protéger contre l'attaque SPA avec l'algorithme RSA *(utilisant l'algorithme square-and-multiply)*, il suffit d'utiliser l'atomicité du canal auxiliaire pour l'algorithme square-and-multiply:

```
#include <string>
#include <cinttypes>

/**
* Side-channel atomic square-and-multiply algorithm
*
* Efficient method for calculating the power of a number
*
* @param x: Base number
* @param d: Binary expansion of exponent d
* @return y = x^d
*/
uint64_t atomicSquareAndMultiply(uint64_t x, string d) {
  uint64_t r0 = 1;
  uint64_t r1 = x;
  int i = 0;
  int k = 0;

  while (i <= (d.length() - 1)) {
    r0 = r0 * (k == 0 ? r0 : r1);
    k = k ^ (d[i] - '0'); // ord("0") = 48
    i += !k;
  }
```

```
24
25   return r0;
26 }
```

Listing 2: Algorithme atomique pour square-and-multiply [34]

Nous observons qu'à chaque itération de la boucle, le même nombre d'opérations est exécuté, ce qui rend l'attaque SPA impossible contre cette implémentation de l'algorithme square-and-multiply.

3.3 Hiding

Les systèmes cryptographiques qui sont protégés par le hiding exécutent les algorithmes cryptographiques de la même manière qu'un système qui n'est pas protégé[14].

La méthode de hiding rompt le lien entre la consommation d'énergie des appareils et les valeurs des données traitées.
Pour cela, il y a deux approches possibles:

1. Construire un système de manière à ce que la consommation d'énergie soit aléatoire.
    - Chaque cycle d'horloge consomme une quantité aléatoire d'énergie.
2. Construire un système qui consomme une quantité égale d'énergie pour toutes les opérations et pour toutes les valeurs de données[15].
    - Des quantités égales d'énergie sont consommées à chaque cycle d'horloge.

La première approche affecte la **dimension temporelle** de la consommation d'énergie, là où la seconde affecte sa **dimension d'amplitude**.

3.3.1 Dimension temporelle

Pour la dimension temporelle, **l'objectif est de réaliser une exécution aléatoire des algorithmes cryptographiques**.

Rappel

Les attaques par DPA nécessitent que les traces enregistrées soient correctement alignées. Cela signifie, que la consommation d'énergie de chaque opération doit être située au même endroit pour chaque trace.

Si cette condition n'est pas remplie, il nous faudra rassembler un nombre considérable de traces supplémentaires.

Ainsi, plus l'exécution d'un algorithme est aléatoire, plus il est difficile d'attaquer le système cryptographique.

[14] Les valeurs intermédiaires sont les mêmes.
[15] Impossible à réaliser dans la pratique, mais nous pouvons nous rapprocher de cet idéal.

Les techniques les plus couramment utilisées pour rendre aléatoire l'exécution des algorithmes cryptographiques sont l'**insertion aléatoire d'opérations factices** et le **mélange** *(shuffling)* des opérations.

Insertion Aléatoire d'Opérations Factices

Le fonctionnement de cette contre-mesure est plutôt simple. Chaque fois qu'un algorithme est exécuté, **un nombre aléatoire est utilisé pour décider du nombre d'opérations factices à insérer à différentes positions**.

Les insertions d'opérations factices sont faites, *avant*, *pendant* et *après* l'exécution de l'algorithme cryptographique.

Il est important que le nombre total d'opérations insérées soit égal à travers toutes les exécutions de l'algorithme. Ainsi, l'attaquant ne peut pas déterminer le nombre d'opérations insérées en mesurant le temps d'exécution de l'algorithme.

Cependant, la position des différentes opérations factices varie entre les exécutions. De cette manière, plus la position varie, plus la consommation d'énergie semble aléatoire.

Le désavantage de cette contre-mesure est que plus le nombre d'opérations fictives insérées est élevé, plus la vitesse d'exécution de l'algorithme est faible.

C'est pour cela qu'en pratique, un compromis est réalisé pour chaque implémentation.

Mélange

L'idée principale est de **changer de manière aléatoire l'ordre des opérations indépendantes**[16] d'un algorithme cryptographique.

Par exemple, dans le cas d'AES, 16 consultations de la S-box doivent être effectuées à chaque tour. Ces consultations sont indépendantes les unes des autres. Ainsi, nous pouvons les réaliser de manière aléatoire. Pour mélanger ses opérations, il faut générer un nombre aléatoire, qui va être utilisé pour déterminer l'ordre des consultations de la S-box.

L'avantage du mélange est que la vitesse d'exécution de l'algorithme est beaucoup moins impactée qu'avec l'insertion aléatoire d'opérations factices.

Cependant, cette méthode ne peut être appliquée qu'à un nombre limité d'opérations d'un algorithme cryptographique. Ce nombre dépend de l'algorithme et de l'architecture de l'implémentation.

Information

En pratique, les insertions aléatoires d'opérations factices et le mélange sont souvent combinés.

[16] Opérations qui peuvent être exécutées dans n'importe quel ordre.

3.3.2 Dimension d'amplitude

Pour la dimension d'amplitude, l'objectif est de **modifier directement les caractéristiques de consommation d'énergie des opérations effectuées**.

Pour réaliser cela, les techniques utilisées vont réduire les fuites d'un dispositif cryptographique en abaissant le **rapport signal sur bruit** *(SNR)* des opérations effectuées.

Definition 3.2 (rapport signal sur bruit) *Rapport entre le signal et le bruit d'une mesure [3].*

$$SNR = \frac{Var(Signal)}{Var(Noise)} = \frac{Var(P_{exp})}{Var(P_{sw.noise} + P_{el.noise})}$$

L'objectif est d'avoir le SNR égal à 0[17]. Pour se rapprocher de cette objectif, deux méthodes peuvent être utilisées:

- Fixer $Var(P_{exp}) = 0$ ***(réduire le signal)***;
  - L'objectif est de construire un appareil dont toutes les opérations nécessitent la **même quantité d'énergie** pour toutes les entrées de données.
- Augmenter $Var(P_{sw.noise} + P_{el.noise})$ jusqu'à l'infini ***(augmentation du bruit)***.
  - L'objectif est de construire un appareil dont la consommation d'énergie est dominée par une **activité aléatoire qui génère du bruit**.

Réduire le signal

Rappel

Les attaques DPA utilisent un grand nombre de trace d'énergie, leur permettant d'exploiter une différence minime dans la consommation d'énergie.

Il faut donc que la contre-mesure appliqué rende la **consommation d'énergie d'un appareil exactement la même pour toutes les opérations et toutes les valeurs de données** pour obtenir une protection parfaite contre les attaques DPA.

Ainsi, réduire le signal n'est pas une tâche trivial.

Il y a deux méthodes pour réduire le signal:

- Utiliser des styles logiques spécifiques pour les cellules des dispositifs cryptographiques;
- Filtrer la consommation électrique d'un système cryptographique.
  - L'idée principale est de retirer toutes les dépendances liées aux données et aux opérations qui influent sur la consommation d'énergie.

[17] Dans la vraie vie, aucune des deux approches que nous verrons ne permet d'atteindre l'objectif idéal de $SNR = 0$. Mais elles permettent de s'en rapprocher.

Augmentation du bruit

La manière la plus simple d'augmenter le bruit est de **réaliser plusieurs opérations indépendantes en parallèle**[18].

Une autre manière d'augmenter le bruit est d'utiliser des générateurs de bruit. Cela va faire augmenter le $Var(P_{sw.noise})$, ce qui diminuera la valeur de SNR

Definition 3.3 (générateurs de bruit) *Les générateurs de bruit effectuent des activités aléatoires parallèlement aux opérations réelles [3].*

	Consommation d'énergie égale	Consommation électrique aléatoire
Dimension temporelle	-	Insertion aléatoire d'opérations factices; Mélange
Dimension d'amplitude	Réduction du signal	Augmentation du bruit

Table 3: Résumé des méthodes de Hiding permettant d'avoir une consommation d'énergie égale ou aléatoire

Information

Sauf pour la méthode de mélange *(shuffling)*, toutes les contre-mesures utilisées pour faire du hiding sont indépendantes des algorithmes cryptographiques et des protocoles qui sont implémentés sur le système cryptographique.

3.3.3 Hiding: Protection logicielle

Étant donné que les caractéristiques de consommation d'énergie des instructions exécutées sur un appareil sont définies par la partie matérielle. Les méthodes de hiding implémentées du point de vue logiciel, afin d'altérer la trace de consommation d'énergie d'un système cryptographique, sont limitées.

Dimension temporelle

La méthode la plus commune pour implémenter du hiding d'un point de vue logiciel consiste à **randomiser l'exécution des algorithmes**[19].

Pour cela, il faut utiliser un générateur de nombres aléatoires sur le système cryptographique. Nous pouvons donc également essayer d'observer le générateur de nombres aléatoires. C'est

[18] Par conséquent, les architectures matérielles des algorithmes cryptographiques avec un bus de données large sont meilleures que les architectures avec un bus de données petit.

[19] Utilisation des insertions aléatoires d'opérations factices et du mélange.

pour cela qu'en général cette méthode ne fournit pas un niveau élevé de protection contre les attaques par analyse de puissance.

Dimension d'amplitude

Les protections concernant la dimension d'amplitude peuvent être mises en place d'un point de vue logiciel, en réordonnant les instructions de l'algorithme cryptographique.

La protection logicielle dans cette dimension permet de se protéger des attaques SPA, mais n'est généralement pas suffisante pour se protéger contre les attaques DPA.

Il existe de nombreuses méthodes pour faire cela:

- **Le choix des instructions:** En générale, toutes les instructions d'un système cryptographique ne divulguent pas le même nombre d'information. C'est pour cela qu'il faut ***choisir avec précaution les instructions utilisées pour implémenter un algorithme cryptographique***. Afin de divulguer le minimum d'information;
- **Le déroulement du programme:** Les sauts conditionnels ainsi que les séquences d'instructions répétées, peuvent être facilement détecté lors d'une observation visuelle. C'est pour cela que ***le développeur doit s'assurer d'éviter les sauts conditionnels qui dépendent de la clé ou de donnée se rapportant à celle-ci***.
- **Adresses mémoires:** Les adresses mémoires utilisées ne doivent pas dépendre de la clé. S'il est nécessaire d'utiliser des adresses dépendant de la clé, il convient de n'**utiliser que des adresses qui laissent échapper des informations similaires**[20].
- **Activité parallèle:** Le développeur peut augmenter le bruit en exécutant des programmes parallèles en même temps que l'algorithme cryptographique.

3.3.4 Hiding: Protection matérielle

Contrairement aux protections logicielles, l'implémentation de hiding en utilisant la couche matérielle offre beaucoup plus d'options pour protéger un système cryptographique.

Dimension temporelle

Il existe deux types de protections matérielles pour la dimension temporelle. Le premier type est similaire aux protections évoquées précédemment, alors que le second type **modifie aléatoirement le signal d'horloge** afin de rendre plus difficile l'alignement des traces de consommation énergétique.

Dans le premier type de protection, nous retrouvons:

- L'**insertion aléatoire d'opérations factices** et le **mélange**.

[20] Par exemple, les adresses de mémoire ayant le même Poids de Hamming doivent être utilisées si l'appareil divulgue le Poids de Hamming des adresses.

- **Insertion aléatoire de cycles fictifs:** C'est une variante de la protection évoquée ci-dessus. Là où l'insertion aléatoire d'opérations factices prend plusieurs cycles d'horloge. **Cette méthode n'insère que des cycles d'horloge individuels.** Pour réaliser cela, les registres du dispositif cryptographique sont généralement dupliqués. Les registres originaux sont utilisés pour stocker les valeurs intermédiaires de l'algorithme cryptographique exécuté. Les autres registres sont utilisés pour stocker des valeurs aléatoires et ainsi insérer des cycles fictifs.

Dans le second type de protection, qui affecte directement le signal de l'horloge, nous avons les protections suivantes:

- **Ignorer une impulsion d'horloge:** Pour cela, il faut filtrer le signal de l'horloge, afin d'ignorer de manière aléatoire les impulsions de celui-ci.
- **Changer la fréquence de l'horloge de manière aléatoire:** Cette fois, nous allons générer un signal d'horloge avec une fréquence qui changera de manière aléatoire. Cela peut être fait en contrôlant la fréquence interne d'un oscillateur grâce à un nombre aléatoire.
- **Utiliser plusieurs domaines d'horloge:** Dans ce cas, plusieurs signaux d'horloges sont générés sur le système cryptographique. L'alignement des opérations de l'algorithme cryptographique est détruit par le changement aléatoire des différents signaux d'horloge.

Information

Toutes les contre-mesures qui affectent la dimension temporelle ne doivent pas être détectables par l'attaquant. Cela signifie que l'attaquant ne doit pas être capable de détecter l'insertion aléatoire d'opération factices ou de cycles, le mélange ou la manipulation du signal d'horloge.

Dimension d'amplitude

Au niveau matériel, la consommation d'énergie d'un système cryptographique peut être rendue égale pour chaque opération et données grâce à **filtrage**. Il est également possible d'**ajouter du bruit aux traces de consommations d'énergie**, afin d'empêcher les attaques par analyses de consommation d'énergie.

Pour la première méthode *(**filtrage**)*, l'objectif est de supprimer toute potentielle source d'exploitation de la consommation d'énergie. Pour cela, la consommation d'énergie est filtrée en utilisant des condensateurs commutés, des sources de courant constant et tout autre type de circuit régulant la consommation d'énergie.

La seconde méthode consiste à ajouter du bruit aux traces de consommations d'énergie. Pour cela, nous allons utiliser des générateurs de bruit. Sachant que les générateurs de bruit sont basés sur la génération d'un nombre aléatoire. Ainsi, pour avoir un impact important sur

la consommation d'énergie, les générateurs de nombres aléatoires doivent être connecté à un réseau de grands condensateurs. La charge et la décharge aléatoires de ce réseau entraînent un bruit dans la consommation d'énergie qui rend les attaques par analyse de consommation d'énergie plus difficiles.

Information

Lorsqu'on implémente du hiding au niveau de la couche matérielle, il est important de savoir que le SNR ne dépend pas uniquement du système cryptographique, mais aussi de la configuration de mesure utilisée pour l'attaque.

Ainsi, le filtrage de la consommation d'énergie rend les attaques plus difficiles si la consommation d'énergie est mesurée au moyen d'une résistance. Cependant, si l'attaquant mesure l'émanation électromagnétique, il n'est pas fortement affecté par cette contre-mesure. C'est pour cela, que les générateurs de bruit doivent être répartis sur l'ensemble de l'appareil au lieu d'être placés dans un seul coin de celui-ci.

3.3.5 Hiding: Protection au niveau des cellules logiques

Definition 3.4 (Cellule logique) *La cellule logique est responsable de l'exécution des instructions et des opérations logiques.*

Elle est généralement représentée comme cela :

- ***Entrées:*** *les registres d'entrée qui fournissent les opérandes pour les opérations;*
- ***UAL:*** *l'unité arithmétique et logique qui exécute les opérations;*
- ***Sorties:*** *les registres de sortie qui stockent les résultats des opérations.*

Réaliser du hiding au niveau des cellules signifie que les cellules logiques d'un circuit sont mises en œuvre de manière à ce que leur consommation d'énergie soit indépendante des données traitées et des opérations effectuées.

Pour mettre en place cette protection, la consommation d'énergie des cellules logiques est constante à chaque cycle d'horloge pour toutes les valeurs logiques traitées.

Une conséquence de ce comportement est que les cellules logiques vont tout le temps consommer la quantité d'énergie maximale pour chaque cycle d'horloge. Ce qui rend la consommation du système cryptographique constante, et par conséquent indépendante des données traitées et des opérations effectuées.

Pour obtenir une cellule logique avec une consommation d'énergie constante, un style logique **dual-rail precharge** (DRP) est souvent utilisé.

Dual-rail (DR)

Contrairement au **single-rail** *(SR)* où le signal logique est transporté sur un seul fil, les cellules DR utilisent une paire de fils pour faire cela.

En logique DR, un fil transporte le signal non inversé a, tandis que l'autre transporte le signal inversé[21] $\bar{a}$ [22]. Ainsi, un signal logique est valide lorsque les deux fils transportent des valeurs complémentaires[23].

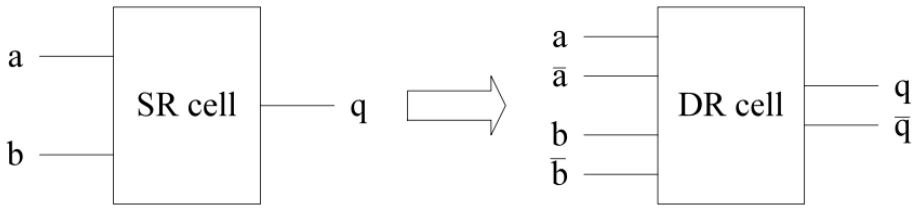


Figure 18: 2 entrées cellules SR et DR [3]

3.4 Masque

Definition 3.5 (masking) *Le masque permet d'avoir une consommation d'énergie indépendante des valeurs intermédiaires calculées pendant l'exécution d'un algorithme cryptographique, même si le système a une consommation d'énergie qui dépend des données.*

Pour faire cela, le masque randomise les valeurs intermédiaires traitées par le dispositif cryptographique [3]

L'idée principale du masque est de définir une **valeur intermédiaire masquée** v_m, correspondante à la valeur intermédiaire v, cachée par une valeur aléatoire m. Ainsi, $v_m = v * m$ [24]. L'attaquant ne connaît pas la valeur aléatoire.

Généralement, les masques sont directement appliqués au texte clair *(*P*)* ou à la clé *(*K*)*. Par conséquent, l'implémentation de l'algorithme doit être légèrement modifiée afin d'appliquer le masque sur les valeurs intermédiaires et de mémoriser celui-ci.

On note que le résultat du chiffrement a également un masque. Il faut donc le retirer à la fin de l'algorithme, pour obtenir le chiffré.

Il est important que toutes les valeurs intermédiaires soient masquées à tout moment de l'exécution. Par exemple, si deux valeurs intermédiaires masquées utilisent le ou-exclusif $\oplus$, il faut s'assurer que le résultat est masqué également. C'est pour cette raison qu'on utilise régulièrement plusieurs masques.

[21] Aussi appelé le **complémentaire**.

[22] Ce type de codage est connu sous le nom de **codage différentiel.**

[23] $(0, 1)$ signifie que le file transportant le signal non inversé est défini à 0 et son complémentaire à 1.

[24] L'opération $*$ est définie en fonction de l'algorithme cryptographique utilisé. Par exemple: $+, \times$ ou $\oplus$.

Cependant, il n'est pas conseillé d'utiliser des nouveaux masques pour chaque valeur intermédiaire, car, le nombre de masques impacte les performances[25].

Information

Le masking est l'une des contre-mesures qui a été et est le plus étudié dans la communauté scientifique. Par conséquent, de nombreux papiers ont été publiés, présentant différents types schémas de masking[a].

Certains schémas possèdent même des preuves de sécurités, ce qui rend leur implémentation très intéressante.

[a] Nous n'étudierons pas tous ses schémas dans ce projet.

On distingue deux types de masques, les **masques booléens** et **arithmétiques**.

3.4.1 Masque Booléen

Dans le masque booléen, les valeurs intermédiaires sont dissimulées en les réalisant un ou-exclusif *($\oplus$)* avec le masque: $v_m = v \oplus m$.

Le masque booléen est très pratique pour des fonctions linéaires,[26] car il est facilement calculable. Cependant, il est beaucoup plus compliqué de l'utiliser sur des fonctions non-linéaires.

3.4.2 Masque Arithmétique

Dans le masque arithmétique, les valeurs intermédiaires sont dissimulées en réalisant une addition ou une multiplication modulaire:

- Addition: $v_m = v + m (\text{mod } n)$
- Multiplication: $v_m = v \times m (\text{mod } n)$.

Information

Le désavantage principal des masques multiplicatifs est qu'ils ne peuvent pas dissimuler la valeur intermédiaire 0.

3.4.3 Partage du secret

Lors de l'application d'un masque, la valeur intermédiaire v est représentée par deux valeurs partagées (v_m, m).

Connaître une des valeurs partagées ne nous révèle aucune information sur v. Cependant, connaître les deux valeurs partagées nous permet de déterminer v.

[25] Elles sont diminuées. C'est pour cela qu'il faut choisir avec précaution le nombre de masques pour avoir une performance acceptable.

[26] $f(x * y) : f(x) * f(y)$

Information

Par conséquent, le masquage correspond à un système de partage du secret qui utilise deux valeurs partagées.

3.4.4 Ordre

Definition 3.6 (Masque d'ordre n) *Un masque d'ordre n signifie qu'on utilise n variables aléatoires pour masquer les calculs. Cela permet de se protéger jusqu'à une attaque DPA d'ordre n-th.*

L'application de plusieurs masques à une valeur intermédiaire et, par conséquent, le suivi de plusieurs valeurs partagées, augmente le coût de l'implémentation.

Il faut utiliser plus de mémoire pour stocker les valeurs partagées, et plus de temps de calcul pour les calculer.

C'est pour cela qu'en pratique les contre-mesures pour des attaques DPA d'ordre supérieur mélangent hiding et masking.

3.4.5 Exemple de masque avec RSA

Dans la plupart des algorithmes de chiffrement asymétrique, les masques arithmétiques sont un bon choix. Utiliser un masque arithmétique pour un chiffrement asymétrique est appelé **blinding**.

Message blinding

Lors de l'opération de déchiffrement, nous pouvons appliquer un masque multiplicatif au chiffré v: $v_m = v \times m^e$. Le résultat v_d peut facilement être retrouvé à la fin de l'algorithme, car on a: $(v_m)^d \equiv (v^d \times m) \mod n$.

Cette contre-mesure, fait que toutes les tentatives d'observation de l'attaquant auront des secrets/exposants différents [35].

Exponent blinding

Cette fois nous allons appliquer un masque additif directement à l'exposant d: $d_m = d + m \times \phi(n)$. On peut facilement enlever le masque du résultat, car: $v^{d_m} \equiv v^d \mod n$.

Cette fois, la contre-mesure rend toutes les valeurs intermédiaires indépendantes de l'entrée de l'algorithme [35].

4 Conclusion

Pour conclure, les attaques par analyse de consommation sont vastes et nous ne faisons qu'une introduction dans ce rapport en présentant les différentes techniques de base. Mais

des attaques et contre-mesures beaucoup plus complexes comme les attaques DPA du second ordre ou les masques d'ordre second ou plus peuvent être réalisées en se basant sur les techniques présentées dans ce rapport.

Lors du second semestre, nous allons réaliser la partie développement du projet, en implémentant les différentes techniques vues dans ce rapport:

- Attaque SPA pour RSA et AES;
- Attaque DPA pour RSA et AES;
- Attaque CPA pour AES;
- Exploration de la base de données ASCAD et utilisation des différents modèles du papier;
- Atomicité du canal auxiliaire pour l'algorithme `square-and-multiply` *(RSA)*;
- Hiding sur AES;
- Masquage AES;
- Masquage RSA *(Message blinding, exponent blinding)*.

Comme pour ce rapport, si nous avons le temps, nous explorerons des attaques et contre-mesures plus complexes lors du prochain semestre.

5 Annexe

```cpp
#include <string>
#include <algorithm>
#include <cinttypes>

using namespace std;

string intToBinary(uint64_t e) {
    if (e == 0)
      return "0";

    string binary;
    while (e > 0) {
        binary.push_back('0' + (e & 1));
        e >>= 1;
    }

    reverse(binary.begin(), binary.end());
    return binary;
  }
```

Listing 3: Expansion binaire d'un nombre

Acronyms

AES Advanced Encryption Standard. 3, 6, 7, 18

CHES Conference on Cryptographic Hardware and Embedded Systems. 8, 19

CNN Convolutional Neural Network. 9

CPA Correlation power analysis. 3, 8, 19–21, 36

DES Data Encryption Standard. 10

DL Deep Learning. 9, 21

DPA Differential power analysis. 3, 8, 14–17, 20, 28, 30, 35, 36

DR Dual-rail. 33

DRP Dual-rail precharge. 32

GF(2^8) Champ Galois d'ordre 2^8. 7

IA Intelligence Artificielle. 8, 21

ML Machine Learning. 8, 9, 21

MSB Most Significant Bit. 19

NIST National Institute of Standards and Technology. 3

RF Random Forest. 9

RSA Rivest–Shamir–Adleman. 7, 17

S-box Rijndael S-box. 7

SCA Side-Channel Attacks. 3, 8, 21

SNR Signal-to-noise ratio - *Rapport signal sur bruit*. 28, 29

SPA Simple power analysis. 3, 8, 14, 36

SR Single-rail. 33

SVM Support Vector Machine. 9

Glossary

AddRoundKey Chaque octet de l'état est combiné avec l'octet correspondant de la clé de ronde. 4–6

Atomicité du canal auxiliaire L'atomicité du canal auxiliaire fait référence à la propriété d'un algorithme ou d'une implémentation cryptographique qui garantit l'exécution d'une opération unique et indivisible (atomicité) sans fuite d'informations sensibles par des canaux involontaires [34]. 25

Attaque par force brute Une attaque par force brute *(bruteforce attack)* consiste à tester, l'une après l'autre, chaque combinaison possible d'un mot de passe ou d'une clé pour un identifiant donné afin se connecter au service ciblé. [36] . 21

Attaques par analyse de consommation électrique Les attaques par analyse de consommation électrique exploitent le fait que la consommation électrique instantanée d'un dispositif cryptographique dépend des données qu'il traite et des opérations qu'il effectue [3]. 3, 8

Attaques par canaux auxiliaires Attaques passives et non-invasives où l'attaquant va observer et mesurer les caractéristiques analogiques de l'implémentation logicielle ou matérielle d'un algorithme de chiffrement, afin d'extraire la clé de chiffrement. Les principales caractéristiques analogiques utilisées en SCA sont le **temps d'exécution**, la **consommation électrique** et les **émissions électromagnétiques**. [2] [3]. 3, 8

Attaques par canaux auxiliaires avec profilage Une attaque par profilage consiste en deux étapes. Premièrement, l'adversaire se procure une copie du matériel cible et définit les fuites physiques. Deuxièmement, il réalise l'attaque sur la cible, afin de retrouver la clé [14] . 8, 21

Attaques par canaux auxiliaires sans profilage L'attaquant a uniquement accès aux fuites physiques capturées sur la machine cible. Pour retrouver la clé secrète utilisée, il va utiliser l'analyse statistique pour détecter les dépendances entre les fuites mesurées et les données sensibles [14] . 8

Attaques par profilage Attaque où un attaquant crée un "profil" d'un dispositif sensible et applique ce profil pour trouver rapidement la clé secrète d'une victime. 11

Coefficient de Corrélation de Pearson Le coefficient de Pearson est un indice reflétant une relation linéaire entre deux variables continues [13, 28, 29]

$$\rho_{WH} = \frac{\mathsf{cov}(W,H)}{\sigma_w \sigma_H} = \frac{E[(W-\mu_W)(H-\mu_H)]}{\sqrt{E[(W-\mu_W)^2]E[(H-\mu_H)^2]}}$$

. 19

Condensateur Composant électronique ou électrique dont l'intérêt de base est de pouvoir recevoir et rendre une charge électrique, dont la valeur est proportionnelle à la tension [37]. 32

Condensateur commuté Un condensateur commuté est un composant électronique ou, plus correctement, un circuit ou un module électronique généralement composé d'un condensateur et de deux commutateurs utilisés pour simuler d'autres composants dans un circuit intégré (CI). La résistance est l'un des composants les plus couramment simulés; les résistances ont tendance à être beaucoup trop grandes et imprécises pour être incorporées dans des circuits intégrés de taille micro [38]. 31

Confusion Volonté de rendre la relation entre la clé de chiffrement et le texte chiffré la plus complexe possible [39]. 5

Differential power analysis Technique exploitant la dépendance de la consommation d'énergie des dispositifs cryptographiques par rapport aux données. Elles utilisent un grand nombre de traces de consommation pour analyser la consommation d'énergie à un moment donné en fonction des données traitées [3]. 14

Diffusion La modification d'un seul bit du texte en clair, change environ la moitié des bits du texte chiffré, et inversement [39]. 4

Distance de Hamming Compte le nombre de transitions de $0 \rightarrow 1$ et $1 \rightarrow 0$ qui ont lieu dans un circuit numérique pendant un certain intervalle de temps [3]. 16

Dual-rail precharge Les styles logiques DRP combinent les concepts de la logique à double rail (DR) et de la logique de précharge. Elles sont utilisés pour les dispositifs cryptographiques afin de rendre la consommation d'énergie des cellules logiques du dispositif constante à chaque cycle d'horloge [3]. 32

Gigue Mesure la variation de la latence au cours du temps. $Gigue = latency(n) - latency(n-1)$, où la mesure $n-1$ est prise à un temps t et n à un temps $t + \delta t$ [40] . 21

Générateurs de bruit Les générateurs de bruit effectuent des activités aléatoires parallèlement aux opérations réelles [3]. 29, 31, 32

Hiding Les systèmes cryptographiques qui sont protégés par le hiding exécutent les algorithmes cryptographiques de la même manière qu'un système qui n'est pas protégé. La méthode de hiding rompt le lien entre la consommation d'énergie des appareils et les valeurs des données traitées [3] . 26, 35

Loi Normale On dit qu'une variable aléatoire X suit la loi normale[21] de paramètres $m \in \mathbb{R}$ et σ^2, avec $\sigma > 0$, ce que l'on note $X \hookrightarrow N(m, \sigma^2)$ si elle est continue et admet pour densité :

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

. 11

Loi Normale Multidimensionnelle Généralisation multidimensionnelle de la loi normale [22].

$$f_{\mu,\sigma}(x) = \frac{1}{(2\pi)^{N/2}\det(\Sigma)^{1/2}} \exp\left[-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right]$$

Avec vecteur $\mu \in \mathbb{R}^N$ représentant son centre et une matrice semi-définie positive $\Sigma \in M_N(\mathbb{R})$ qui est sa matrice de variance-covariance et un vecteur $x \in \mathbb{R}^N$. 11

Masking Le masking permet d'avoir une consommation d'énergie indépendante des valeurs intermédiaires calculées pendant l'exécution d'un algorithme cryptographique, même si le système a une consommation d'énergie qui dépend des données. Pour faire cela, le masking randomise les valeurs intermédiaires traitées par le dispositif cryptographique [3] . 33, 35

MixColumns Opération de mélange linéaire sur les colonnes de l'état. Chaque colonne est transformée en un nouveau vecteur de quatre octets, en combinant les octets de la colonne actuelle pour diffuser les informations entre eux. 5

Poids de Hamming Nombre de bits qui sont à 1 [3]. 16, 30

Rapport signal sur bruit Rapport entre le signal et le bruit d'une mesure [3]

$$SNR = \frac{Var(Signal)}{Var(Noise)} = \frac{Var(P_{exp})}{Var(P_{sw.noise} + P_{el.noise})}$$

. 28

Réseaux de Neurones Convolutifs Ils utilisent des données tridimensionnelles pour les tâches de classification d'images et de reconnaissance d'objets. Ils se distinguent des autres réseaux neuronaux par leurs performances supérieures avec des entrées de signaux d'image, de parole ou audio [32]. 22

ShiftRows Effectue une transposition cyclique des lignes de l'état. La première ligne reste inchangée, la deuxième est décalée d'un octet vers la gauche, la troisième de deux octets, et la quatrième de trois octets. Cela contribue à la confusion en modifiant l'ordre des données. 5, 6

Simple power analysis Technique qui consiste à interpréter directement les mesures de consommation d'énergie recueillies au cours d'opérations cryptographiques [11]. 9

SubBytes Substitution non linéaire où chaque octet de l'état est remplacé par un autre octet selon une table de correspondance (table de substitution ou S-box). Cette substitution renforce la diffusion des données. 4, 6

Template Attacks Les templates attacks exploitent le fait que la consommation énergétique dépend des données qui sont traitées. Dans une attaque basées sur des modèles, nous caractérisons les traces de puissance par une loi normale multidimensionnelle [3]. 11, 12

References

[1] C. O'Flynn, "0x504 attacking aes with power analysis." Available at https://www.youtube.com/watch?v=5Hn2D5lrzVo.

[2] J. H. J.H. Silverman, Jill Pipher, *An Introduction to Mathematical Cryptography*. Springer, 2 ed., 2014.

[3] T. P. Stefan Mangard, Elisabeth Oswald, *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer, 1 ed., 2007.

[4] J. Daemen and V. Rijmen, *The design of Rijndael: AES — the Advanced Encryption Standard*. Springer-Verlag, 2002.

[5] "Specification for the advanced encryption standard (aes)." Federal Information Processing Standards Publication 197, 2001.

[6] J.-P. Aumasson, *Serious Cryptography - A Pratical Introduction to Modern Encryption*. San Francisco: No Starch Press, 1 ed., 2018.

[7] "Advanced encryption standard - wikipedia." Available at https://en.wikipedia.org/wiki/Advanced_Encryption_Standard.

[8] Q. Meunier, "Attaques side-channel, masquage et vérification, tests statistiques, dfa, attaques de caches." Available at https://largo.lip6.fr/~meunier/docs/SCA_slides.pdf.

[9] "Finite field." Available at https://en.wikipedia.org/wiki/Finite_field.

[10] P. C. Kocher, "Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems," in *Advances in Cryptology—CRYPTO'96: 16th Annual International Cryptology Conference Santa Barbara, California, USA August 18–22, 1996 Proceedings 16*, pp. 104–113, Springer, 1996.

[11] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology — CRYPTO' 99* (M. Wiener, ed.), (Berlin, Heidelberg), pp. 388–397, Springer Berlin Heidelberg, 1999.

[12] P. Kocher, J. Horn, A. Fogh, , D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher, M. Schwarz, and Y. Yarom, "Spectre attacks: Exploiting speculative execution," in *40th IEEE Symposium on Security and Privacy (S&P'19)*, 2019.

[13] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *Cryptographic Hardware and Embedded Systems - CHES 2004* (M. Joye and J.-J. Quisquater, eds.), (Berlin, Heidelberg), pp. 16–29, Springer Berlin Heidelberg, 2004.

[14] R. Benadjila, E. Prouff, R. Strullu, E. Cagli, and C. Dumas, "Deep learning for side-channel analysis and introduction to ascad database," *Journal of Cryptographic Engineering*, vol. 10, no. 2, pp. 163–188, 2020.

[15] T. Bartkewitz and K. Lemke-Rust, "Efficient template attacks based on probabilistic multi-class support vector machines," in *Smart Card Research and Advanced Applications: 11th International Conference, CARDIS 2012, Graz, Austria, November 28-30, 2012, Revised Selected Papers 11*, pp. 263–276, Springer, 2013.

[16] A. Heuser and M. Zohner, "Intelligent machine homicide: Breaking cryptographic devices using support vector machines," in *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pp. 249–264, Springer, 2012.

[17] L. Lerman, G. Bontempi, and O. Markowitch, "Power analysis attack: an approach based on machine learning," *International Journal of Applied Cryptography*, vol. 3, no. 2, pp. 97–115, 2014.

[18] M. Randolph and W. Diehl, "Power side-channel attack analysis: A review of 20 years of study for the layman," *Cryptography*, vol. 4, no. 2, p. 15, 2020.

[19] N. Courtois, "All about side channel attacks." Available at `http://www.nicolascourtois.com/papers/sc/sidech_attacks.pdf`.

[20] N. Technology, "Template attacks." Available at `http://wiki.newae.com/Template_Attacks`.

[21] Bibm@th, "Loi normale." Available at `https://www.bibmath.net/dico/index.php?action=affiche&quoi=./l/loinormale.html`.

[22] Wikipedia, "Loi normale multidimensionnelle." Available at `https://fr.wikipedia.org/wiki/Loi_normale_multidimensionnelle`.

[23] K. Schramm, G. Leander, P. Felke, and C. Paar, "A collision-attack on aes: Combining side channel-and differential-attack," in *Cryptographic Hardware and Embedded Systems-CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings 6*, pp. 163–175, Springer, 2004.

[24] BibM@th.net, "Régression linéaire et moindres carrés." Available at `https://www.bibmath.net/dico/index.php?action=affiche&quoi=./r/reglin.html`.

[25] S. Vinatier, "Introduction à la cryptologie."

[26] G. Burghoorn, "Power analysis introductory walkthrough - a case study: Rsa." Available at `https://coastalwhite.github.io/intro-power-analysis/rsa.html`.

[27] K. L. Bert den Boer and G. Wicke, "A dpa attack against the modular reduction within a crt implementation of rsa," 2003.

[28] N. Technology, "Correlation power analysis." Available at `http://wiki.newae.com/Correlation_Power_Analysis`.

[29] B. U. de Liège, "Corrélation de pearson." Available at `http://www.biostat.ulg.ac.be/pages/Site_r/corr_pearson.html`.

[30] B. R. U. M. . (LIRMM), "Attaques par canaux cachÉs side-channel attacks." Available at https://www.lirmm.fr/~rouzeyre/PDF/Crypto/AttaquesCanauxCaches.pdf.

[31] Y.-H. Chou and S.-L. L. Lu, "A high performance, low energy, compact masked 128-bit aes in 22nm cmos technology," in *2019 International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*, pp. 1–4, 2019.

[32] IBM, "Qu'est-ce qu'un convolutional neural networks (cnn) ?." Available at https://www.ibm.com/fr-fr/topics/convolutional-neural-networks.

[33] DataScientest, "Convolutional neural network: Everything you need to know." Available at https://datascientest.com/en/convolutional-neural-network-everything-you-need-to-know.

[34] B. Chevallier-Mames, M. Ciet, and M. Joye, "Low-cost solutions for preventing simple side-channel analysis: Side-channel atomicity," *IEEE Transactions on computers*, vol. 53, no. 6, pp. 760–768, 2004.

[35] A. Bauer, É. Jaulmes, E. Prouff, and J. Wild, "Horizontal and vertical side-channel attacks against secure rsa implementations," in *Cryptographers' Track at the RSA Conference*, pp. 1–17, Springer, 2013.

[36] CNIL, "Force brute (attaque informatique)." Available at https://www.cnil.fr/fr/definition/force-brute-attaque-informatique.

[37] Techno-Science.net, "Physique condensateur (électricité) - définition." Available at https://www.techno-science.net/definition/3148.html.

[38] L. Shenzhen Baiqiancheng Électronique Co., "Qu'est-ce qu'un condensateur commuté?." Available at https://fr.ems-pcbassembly.com/info/company-certificate-44527673.html.

[39] "Confusion and diffusion - wikipedia." Available at https://en.wikipedia.org/wiki/Confusion_and_diffusion.

[40] P.-F. Bonnefoi, "Cours - protocoles et programmation réseau." Available at https://p-fb.net/master1/proto_prog/cours/Cours_PPRes.pdf.