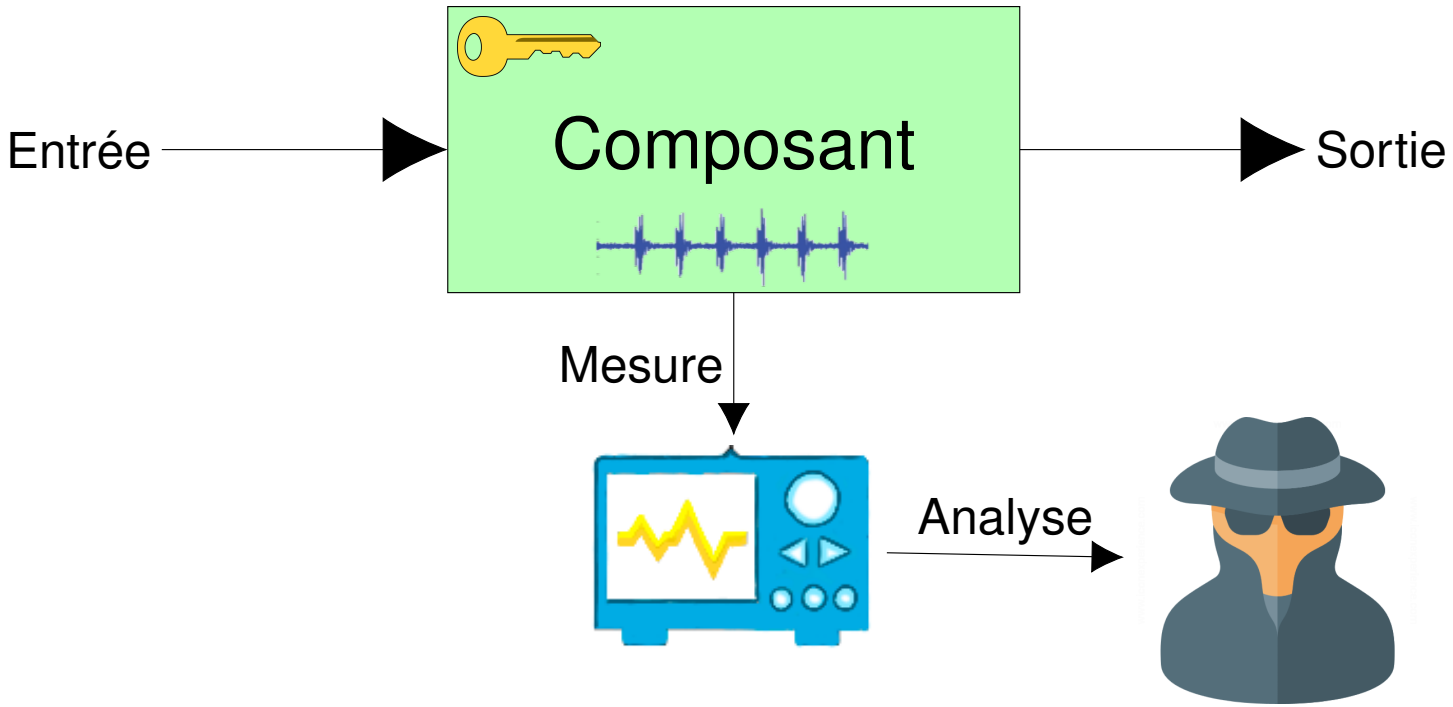
Architecture Matérielle 3^e année
Module Sécurité - Partie 2

Vincent Migliore

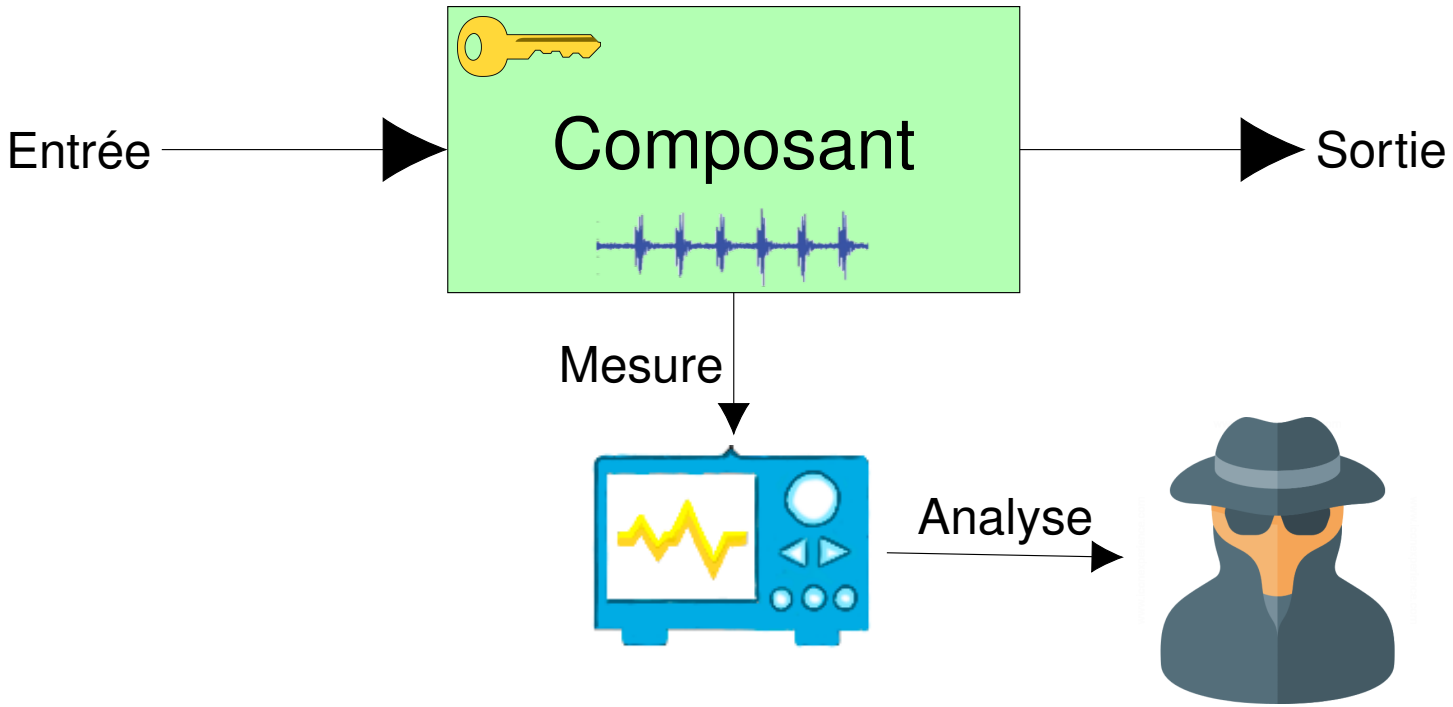
vincent.migliore@insa-toulouse.fr

INSA-TOULOUSE / LAAS-CNRS



Principe

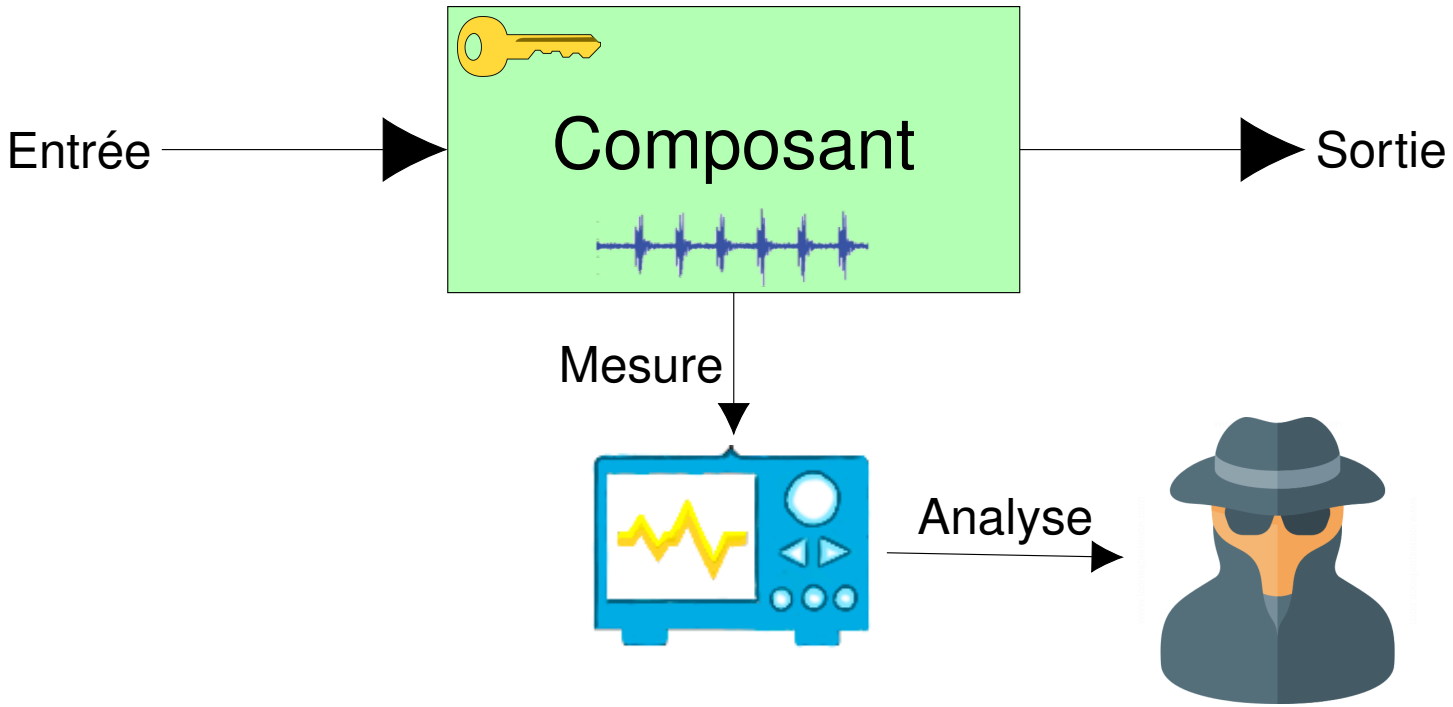
Les attaques matérielles reposant sur l'écoute de la consommation visent à extraire de la mesure de la consommation énergétique de l'information sur l'activité du composant. Elles fonctionnent également par écoute du champ électromagnétique.



Applications classiques

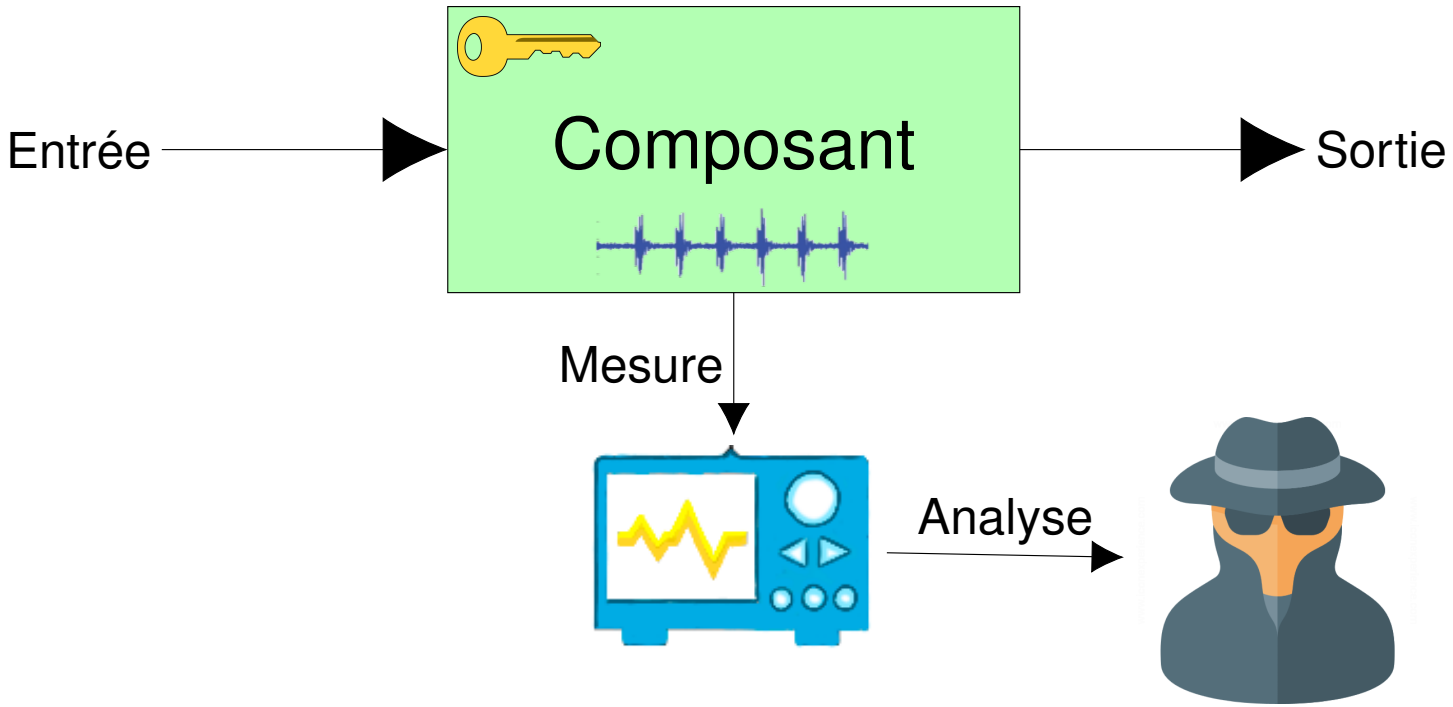
Ces attaquent sont en général utilisées pour extraire des données sensibles, notamment des clés de chiffrement, des mots de passe, etc.. Tout ce qui permet d'obtenir des privilèges supplémentaires, d'accéder à des espaces protégés, à usurper une identité, etc..

Attaques par écoute de la consommation



Avantages

- Non-intrusives : quasiment non détectable.
- Puissantes : elles contournent le modèle d'attaquant classique qui n'écoute que les échanges (messages) avec l'extérieur.
- Difficiles à empêcher : la surface d'attaque correspond à toutes les parties du circuit qui manipule le secret.

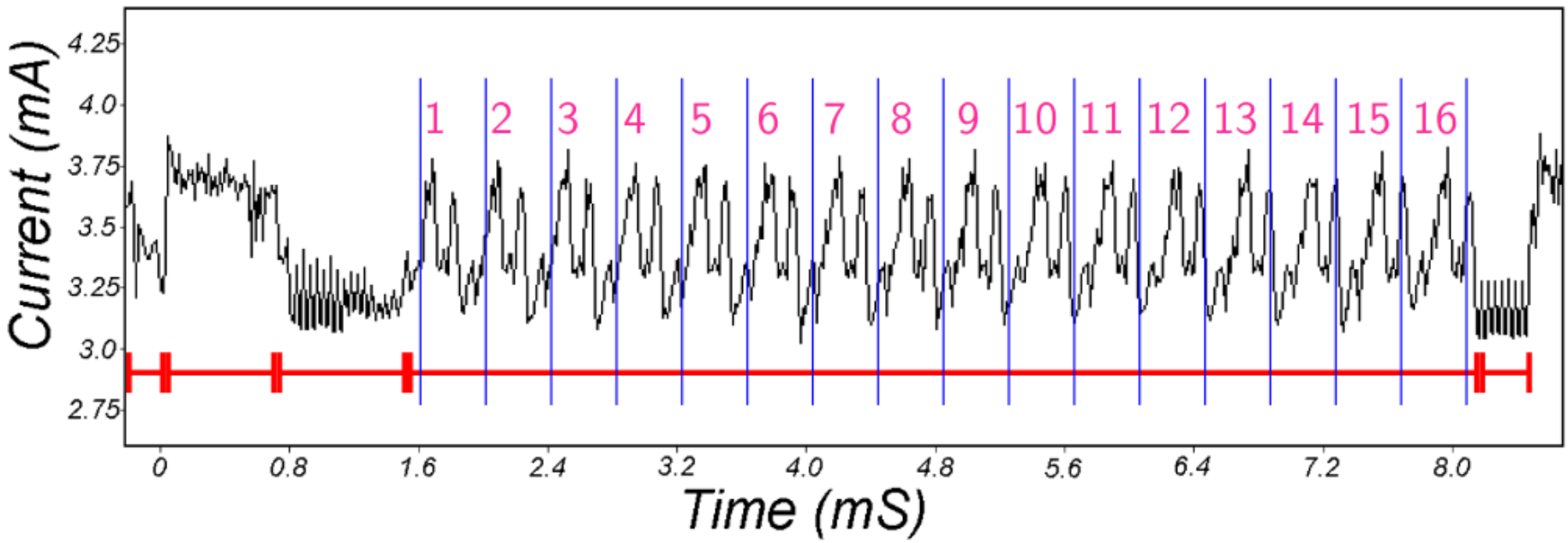


Inconvénients

- Proximité du système : le système doit être à porté de l'attaquant.
- Moins efficace sur les systèmes performants : les fréquences élevées et la présence de nombreux bruits parasites tendent à complexifier l'attaque.

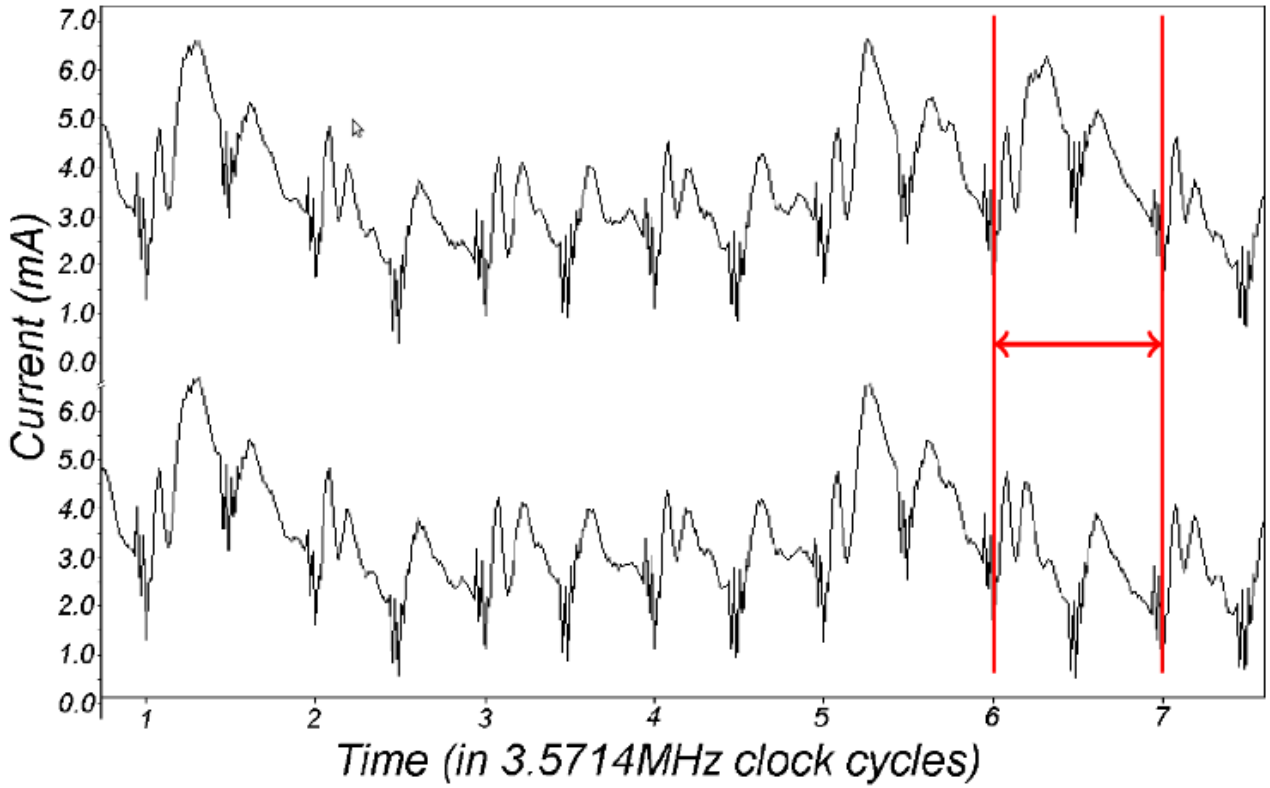
$\rightarrow$ les objets IoT, les systèmes connectés (cartes à puces, système d'authentification dans les bâtiments, dans les transports, etc) sont plus exposés et donc la cible de ces attaques.

Exploitation directe de la consommation : L'attaque SPA



Simple Power Analysis - Attaque sur les implémentations non-temps constant

En principe, pour une question d'optimisation de calcul, le temps d'exécution peut varier en fonction des paramètres d'entrée (arrêt prématuré de boucles, exécution de certaines fonction et pas d'autres, etc ...) amenant un aléa de contrôle. Contrairement à ce que l'on pourrait penser, l'énergie dynamique est clairement identifiable à l'aide d'une sonde de mesure (courant ou EM). Si le temps d'exécution est dépendant du secret, il est trivial pour un attaquant de récupérer le l'information sur le secret.



Simple Power Analysis - Attaque sur les implémentations temps constant

Malheureusement, le temps de constant seul n'est pas suffisant pour sécuriser le calcul. En effet, il existe des différences de consommations perceptibles lorsque les données manipulées varient. L'attaquant doit donc cibler des zones d'intérêt où le secret est manipulé.

Modélisation des fuites

Intéret de la modélisation

La modélisation des fuites est une étape fondamentale dans la compréhension des sources de fuite (que ce soit pour attaquer ou défendre un système). Elle permet, en particulier, de corréler l'algorithme exécuté et la consommation énergétique.

Consommation des transistors MOS : Consommation de transition

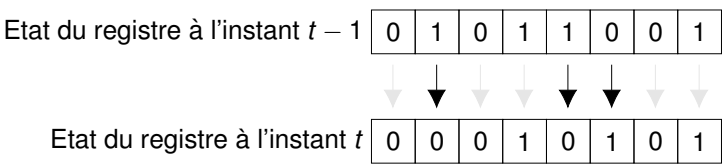
Un modèle classique de la consommation d'un transistor MOS est le suivant :

$$P_T = C_{pd} \times V_{cc}^2 \times f_l \times N_{sw} \tag{1}$$

Avec P_T la puissance consommée, f_l la fréquence du signal d'entrée, V_{cc} la tension d'alimentation, C_{pd} une constante et N_{sw} le nombre de bits ayant changé d'état.

Choix d'un niveau d'abstraction

A partir du modèle précédent, on montre que la consommation dynamique est proportionnelle au nombre de transistors qui ont changés d'état (N_{sw}). Cette échelle est cependant très inadaptée, il est en effet très complexe de lier l'algorithme et la consommation énergétique d'un transistor. Pour les composants séquentiels, on choisi une échelle intermédiaire qui est la consommation au niveau des bascules. Dans le cas des processeurs, on va également plus loin et en modélisant la consommation niveau registre, permettant de faire le lien direct avec l'architecture du jeu d'instructions.

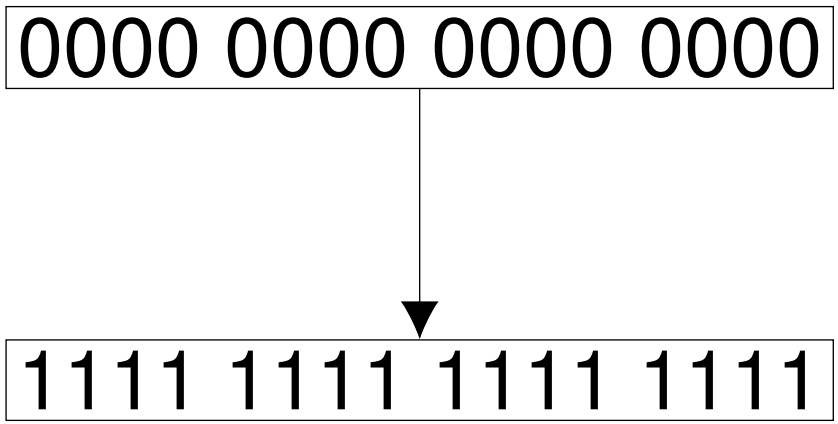


Modèle en distance de Hamming

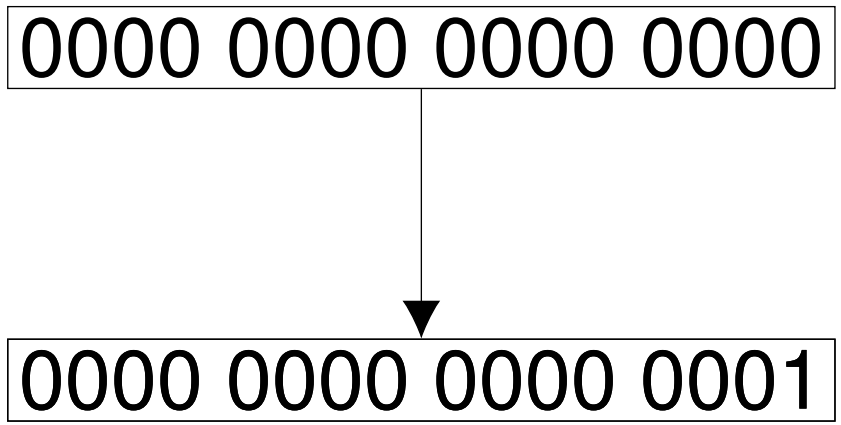
Le modèle en distance de Hamming est un modèle de fuite très classique pour les attaques par écoute de la consommation énergétique. Il estime la fuite à l'instant t comme étant la somme des bits qui ont changés d'état entre t et $t-1$. En prenant l'exemple de la figure de gauche, la distance de Hamming est de 3.
Importante limitation : Ce modèle n'est applicable que si l'on a des informations assez précise sur l'architecture, notamment on doit être en capacité de savoir quelles actions déclenchent des transitions de bit.

Modèle en poids de Hamming

Lorsque le modèle en distance de Hamming n'est pas applicable faute de connaissance suffisante de l'architecture, mais que l'algorithme est connu, une solution de repli consiste à estimer la fuite comme étant le nombre de bits à 1 du résultat de chaque opération de l'algorithme. Par exemple, pour l'opération $a + b = c$, la fuite sera estimée comme le poids de Hamming de c. Le modèle étant imprécis, ce modèle ne décrit pas toutes les fuites possibles de l'implémentation.



16ϵ



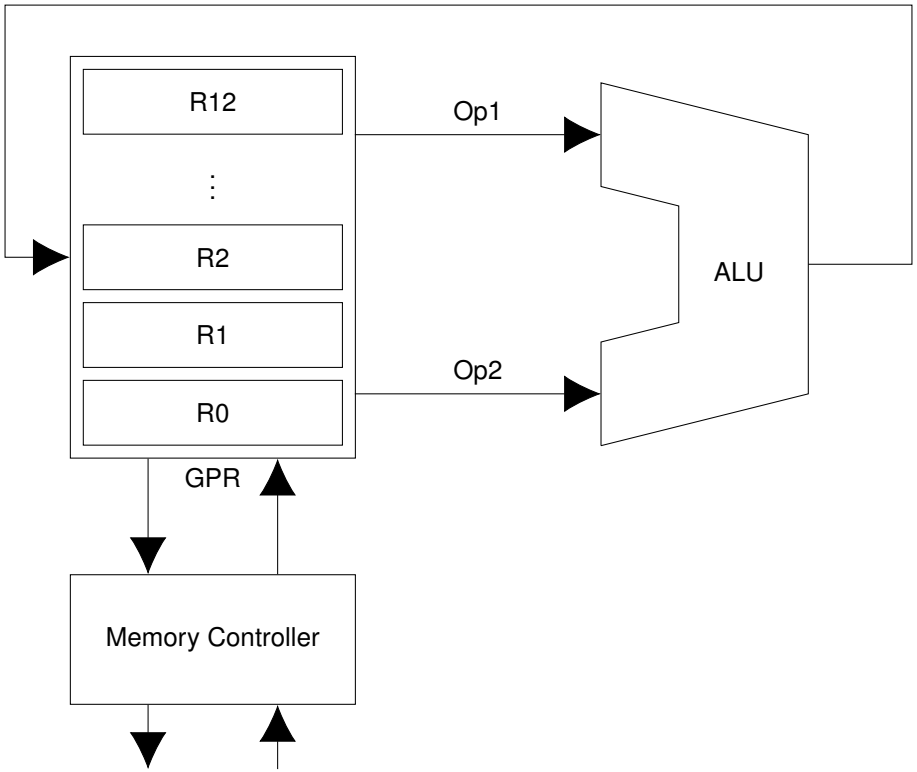
ϵ

Limitations de la SPA

Comme présenté sur le schéma ci-dessus, la différence de consommation dynamique est d'autant plus importante que le nombre de bascules ayant changé d'état est grand. Ainsi, un nombre trop faible de variation peut ne pas être exploitable juste de manière visuelle. Cependant, là encore, chercher à minimiser le nombre de transitions reste très insuffisant comme protection. Il est notamment possible de :

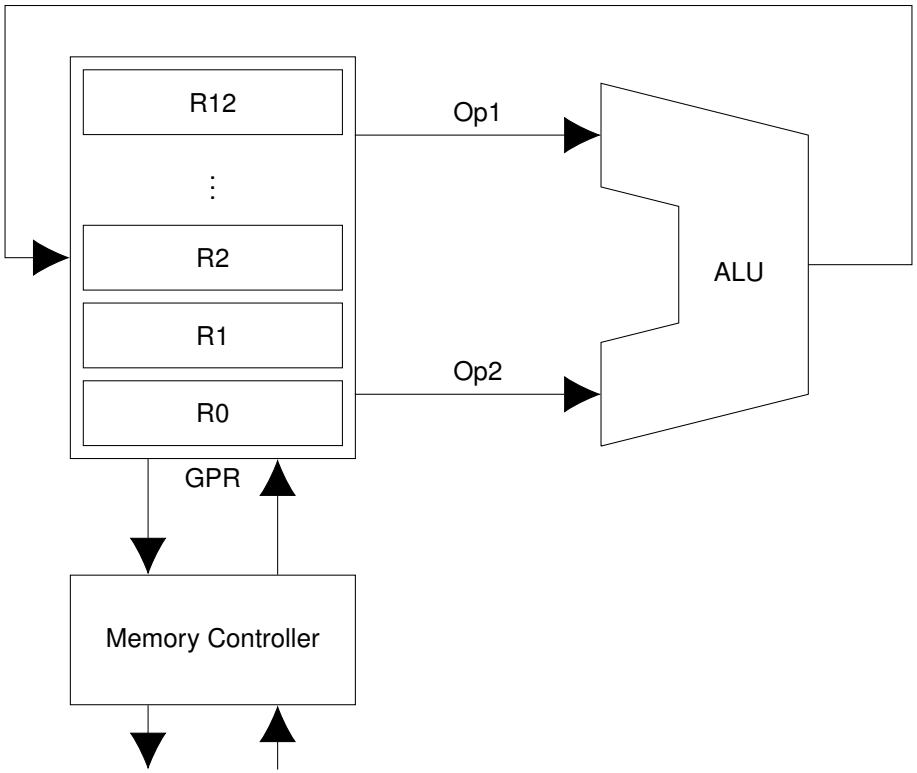
- Capturer d'avantage de trace et de faire des tests statistiques ;
- Créer un modèle du composant pour extraire des informations même provenant de faibles variations (attaques par template).

Exemple du processeur



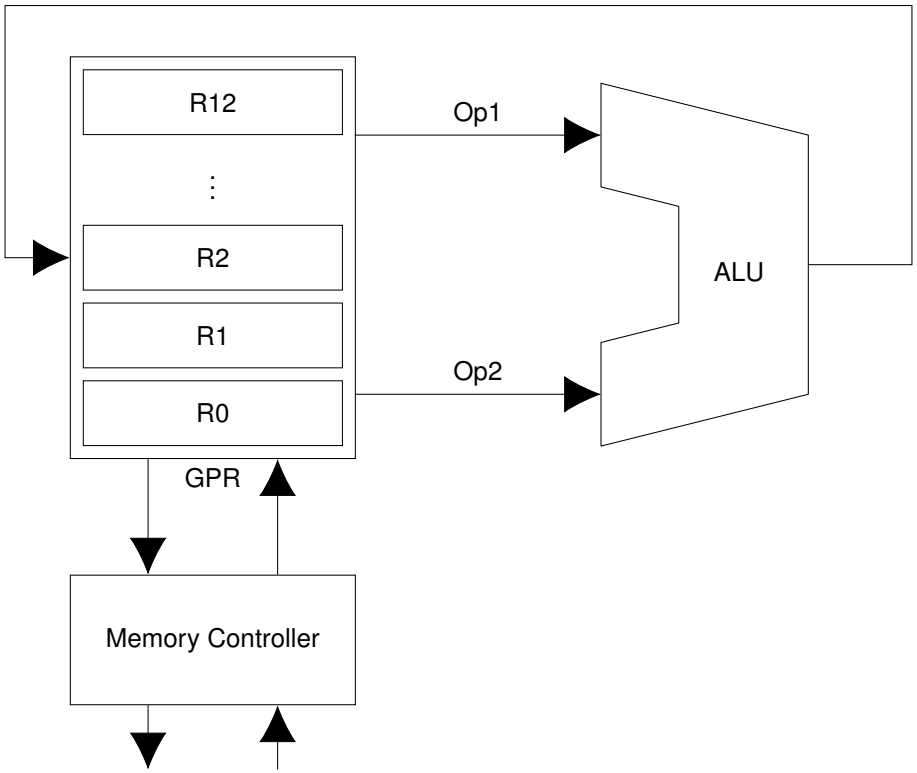
- Dans la processeur, la principale source de fuite provient des registres, qui sont modifiés typiquement :
  - Lors des opérations de lecture en RAM.
  - La copie registre-registre.
  - La copie dans un registre du résultat d'une opération arithmétique et logique.
- Ici, on utilise en principe le modèle en distance de Hamming, l'architecture niveau ISA étant relativement bien maîtrisée.

Exemple du processeur

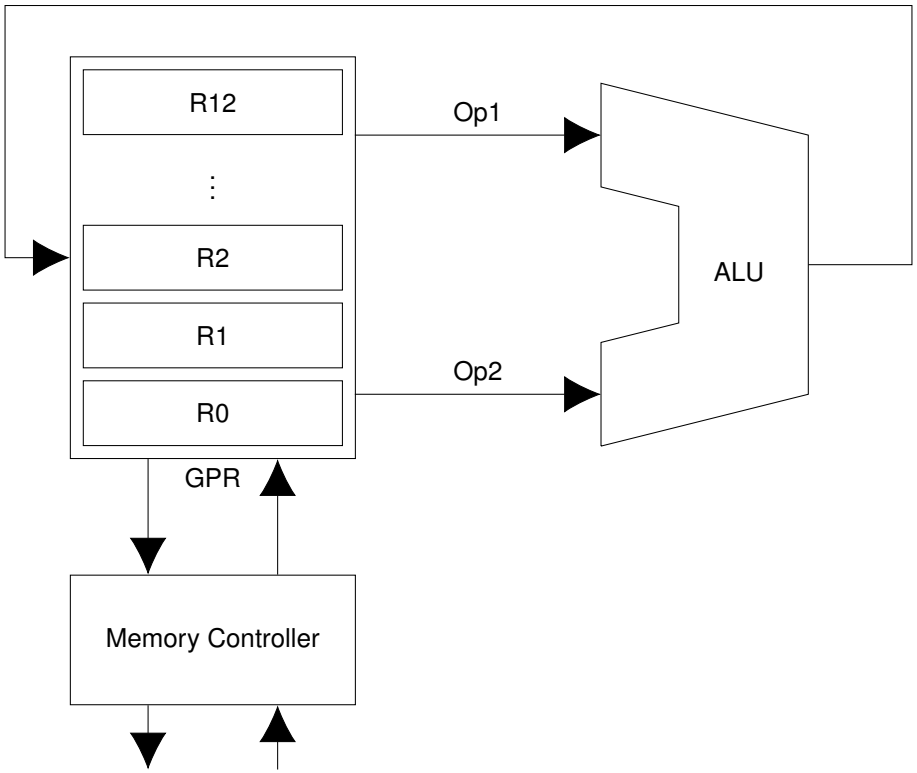


- Example:
  - $R0 = (01001100)_2$
  - $R1 = (00111010)_2$
  - Calcul de la fuite de ADD $R1, R0 =$

Exemple du processeur



- Example:
  - $R0 = (01001100)_2$
  - $R1 = (00111010)_2$
  - Calcul de la fuite de ADD $R1, R0 = 5$



Remarque

Bien qu'en général, les registres généraux font l'objet d'une attention très forte, cependant il ne faut pas négliger la contribution des autres bascules du circuit, en particulier les bascules liées au pipe-line.

Les méthodes offensives

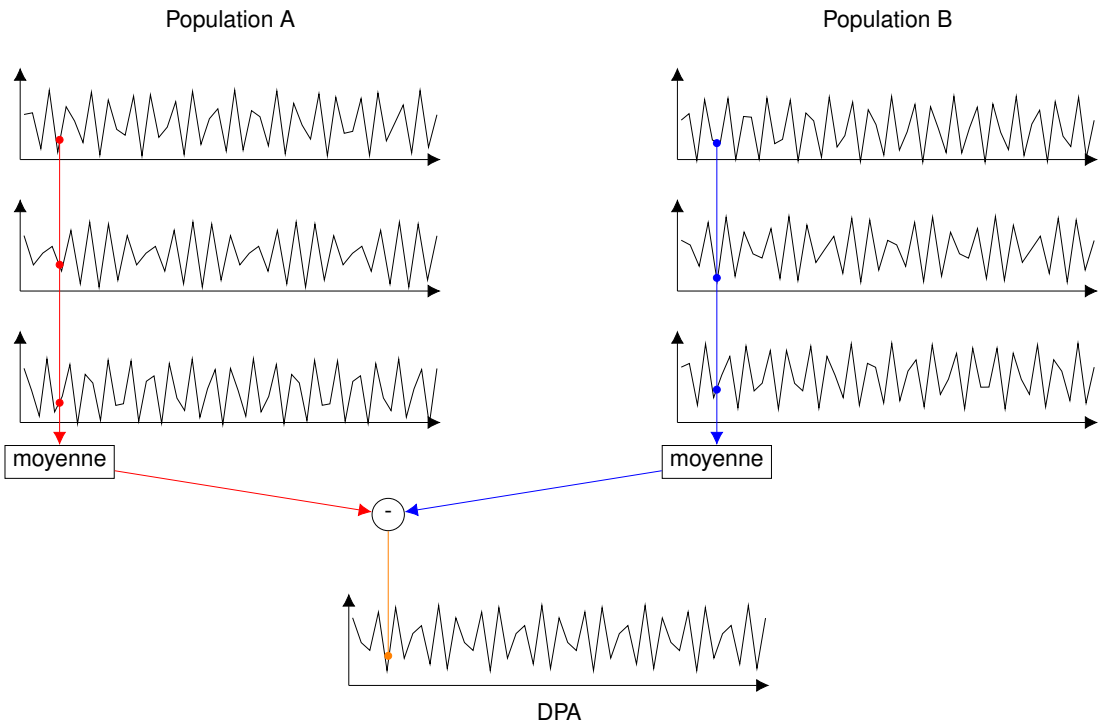
Principes généraux et prérequis

les méthodes offensives réalistes reposent sur des méthodes d'analyse statistique à partir non pas d?une unique trace mais d?un ensemble de traces. Elles demandent un certain nombre de prérequis pour fonctionner efficacement et en particulier il faut :

- connaître l'algorithme et si possible l'architecture ;
- avoir accès à un nombre suffisant de traces de consommation dont le secret qui ne varie pas, mais dont les entrées de l'algorithme analysé varie (entrée contrôlée ou non par l'attaquant) ;
- connaître soit l'entrée, soit la sortie.

Cas typiques où les attaques statistiques fonctionnent bien

- **Les algorithmes de cryptographie** : Ce sont des algorithmes qui manipulent une clé de sécurité pour chiffrer les données qui est renouvelé rarement, voire très rarement en fonction de l'algorithme. De plus la sortie de l'algorithme est par principe accessible par un attaquant (en écoutant le réseau informatique par exemple).
- **Les systèmes d'authentification** : Dans ce type de système, l'attaquant a un certain contrôle sur l'entrée, permettant de choisir des entrées favorables à l'attaque. Comme il les choisit, l'attaquant connaît également l'entrée.



Principe

L'attaque DPA (Differential Power Analysis) repose sur la construction d'une fonction de sélection qui prend en paramètres :

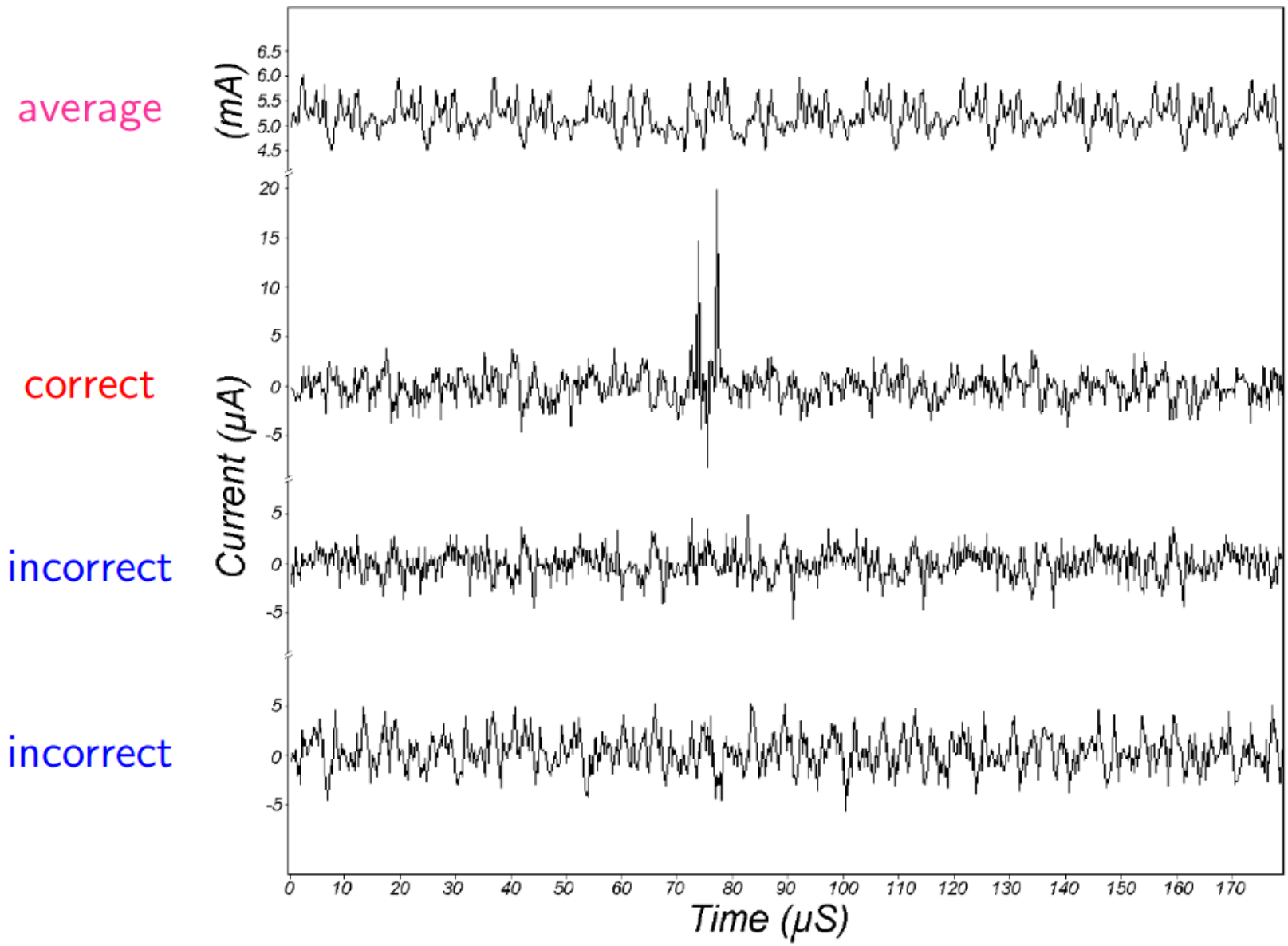
- l'entrée (ou la sortie) utilisée pour chaque trace capturée ;
- une hypothèse sur la valeur de un ou plusieurs bits du secret.

Cette fonction de sélection va permettre de trier les traces de consommation en 2 populations, distinguables statistiquement dans le cas où l'hypothèse sur la clé est juste (on obtient une corrélation). Si l'hypothèse sur le secret est faux, alors les populations ne sont pas distinguables. Pour mener l'attaque, il suffit donc de tester toutes les hypothèses possibles et prendre l'hypothèse amenant la plus grande corrélation.

DPA $\neq$ Force brute

Il ne faut pas confondre DPA et force brute. Dans le cas de la force brute, on va tester tous les secrets possibles, alors qu'avec la DPA, on ne fait de recherche exhaustive que sur quelques bits du secret tout au plus (et de préférence un seul).

Exemple de résultat de DPA



Le choix est relativement simple, il suffit de prendre une variable interne à l'algorithme que l'on peut calculer à l'aide de la valeur de quelques bits de la clé et de l'entrée (ou la sortie) de l'algorithme. La fonction de sélection peut être choisie comme la valeur d'un bit particulier de la variable.

Cas d'un algorithme de chiffrement (cryptographie)

Soit $D(C_i, b, K_s)$ la fonction de décision prenant en paramètres le chiffré C_i (sortie de l'algorithme de chiffrement), b le bit de la variable interne de l'algorithme et K_s les bits de lés associés à b.
Soit $\Delta_D[j]$ la valeur du calcul de la DPA au point j, et $T_i[j]$ la trace de consommation de la i_{eme} trace au point j. La DPA s'obtient en triant les traces en deux populations puis en calculant la moyenne de consommation à chaque échantillon, soit :

$$\Delta_D[j] = \frac{\sum D(C_i, b, K_s) T_i[j]}{\sum D(C_i, b, K_s)} - \frac{\sum (1 - D(C_i, b, K_s)) T_i[j]}{\sum D(C_i, b, K_s)} \tag{2}$$

$$\Delta_D[j] \approx 2\left(\frac{\sum D(C_i, b, K_s) T_i[j]}{\sum D(C_i, b, K_s)} - \frac{\sum T_i[j]}{m}\right) \tag{3}$$

Avec m le nombre de traces.
On peut remarque que :

- Si l'hypothèse de clé est fausse, alors D est décorrélée avec la clé, les traces sont aléatoirement réparties entre les populations $\rightarrow \Delta_D[j] \approx 0$.
- Si l'hypothèse de clé est juste, alors D est corrélée avec la clé, et donc $\Delta_D[j]$ se rapproche de la contribution du bit b à la consommation.

Exemple

Hypothèse juste		Hypothèse fausse	
Population A	Population B	Population A	Population B
0.66	0.12	0.12	0.66
0.66	0.73	0.73	0.66
0.66	0.24	0.24	0.66
0.66	0.28	0.66	0.28
0.66	0.64	0.64	0.66
0.66	0.05	0.66	0.05
0.66	0.92	0.92	0.66
Moy : 0.66	Moy : 0,42	Moy : 0.56	Moy : 0,52

Principe

L'attaque CPA (Correlation Power Analysis) est une attaque plus puissante que l'attaque DPA qui vise à calculer la corrélation entre des traces de consommation mesurées et des traces de consommation estimée via le modèle de fuite. Pour l'estimation des fuites, on choisit une variable interne liée au secret, puis pour chaque trace, on applique le modèle de fuite pour estimer la consommation.

Coefficient de corrélation de Pearson

Le coefficient de corrélation de Pearson est l'algorithme classique pour le calcul de corrélation dans la CPA, il repose sur le calcul de la covariance entre deux variables statistiques :

$$\rho(X, Y) = \frac{Cov(X, Y)}{\sigma_x \sigma_y} \tag{4}$$

où X et Y sont deux variables, $\rho(X, Y)$ le coefficient de corrélation de Pearson des variables X et Y, $Cov(X, Y)$ la covariance des deux variables X et Y; et σ_x et σ_y l'écart type des variables X et Y.
En développant l'équation, on obtient la formule suivante :

$$\rho(X, Y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{5}$$

.

DPA vs CPA

Dans la pratique, on observe une différence d'un facteur 10 environ sur le nombre de traces nécessaires avec d'observer une corrélation, la CPA est donc nettement plus efficace. Petit point positif de la DPA, elle permet d'identifier sur la courbe l'endroit de la fuite (par construction).

Les méthodes défensives

Les techniques de dissimulation

Ces méthodes visent à chercher à pénaliser l'attaquant afin de rendre l'attaque difficile et à le décourager, par exemple en empêchant la synchronisation de l?attaquant avec le système ou en ajoutant du bruit avant, après ou entre les calculs sans modifier l?algorithme lui-même.

Les techniques de masquage

Ces méthodes visent à rendre la fuite de consommation indépendante du secret à partir d?un modèle de fuite (ce qui modifie les calculs effectués par l?algorithme). Elles ont été prouvées formellement.

De nombreuses contremesures de dissimulation ont été proposées dans la littératures (voir ci-dessous)). Bien que considérées plus légères que les contremesures formelles, elles souffrent principalement de leur manque de preuve formelle, qui conduit à deux limitations :

- l?évaluation de la sécurité se fait principalement de manière expérimentale ;
- il y a un risque de découvrir une méthode d?attaque qui réduise voire annule la contremesure.

Réduction de la consommation énergétique

La réduction de la consommation (changement de technologie de transistor par exemple) est une technique de dissimulation, demandant à l'attaquant d'avantage de traces.

Dual techno

Technique visant à implémenter un algorithme et son dual dans le même circuit, afin d'homogénéiser la consommation.

Desynchronisation

Ces techniques sont variées, comme ajout de délais aléatoires, changement de l?ordre des instructions parallélisables, ajout d?instructions inutiles, ...

Ajout de bruit

Augmentation artificielle du niveau de bruit à l'aide de calculs parallèles ou de générateurs de bruits dédiés.

Principe

Le principe du masquage est de rendre la fuite de consommation indépendante du secret, en mélangeant le secret avec un bruit uniforme appelé masque. Ce mélange est effectué classiquement dans un groupe (au sens mathématique) en s?assurant que l?opération sur le groupe soit compatible avec l?arithmétique de l?algorithme qui sera en charge de faire les calculs sur le secret.

Masquage booléen

- Pour un secret s et un masque r, le masquage booléen se calcule par :
  - $s_0 = s \oplus r$
  - $s_1 = r$

  Avec s_0 et s_1 nommés partagés (shares).
- Notons que comme s et indépendant de r, la connaissance de s_0 ne permet pas de remonter à s.

Masquage arithmetique

- Pour un secret s et un masque r, le masquage arithmétique se calcule par :
  - $s_0 = s + r$
  - $s_1 = r$

Différentes techniques ont été proposées pour intégrer et valider des techniques de masquage de manière automatique. Elles reposent en général sur une modélisation abstraite de l?architecture matérielle du système cible couplée à une caractérisation des sources de fuite. Elles restent aujourd'hui réservés aux experts du domaine.

Exemples de méthodes et d'outils'outils

- Intégration du masquage directement durant la phase de compilation (COGITO, CEA) ;
- Vérification automatique du masquage d'un code source (maskVerif, CryptoExperts) ;
- Vérification automatique du masquage d'un code compilé (ARISTI, Lip6).