

# Sécurité et systèmes embarqués

Pablo Rauzy

`pr@up8.edu`

`pablo.rauzy.name/teaching/sese`



UFR MITSIC / M1 informatique

Séance 5

Les canaux auxiliaires

# Les canaux auxiliaires

---

- ▶ Un algorithme cryptographique peut être vu de deux façons :
  - d'un côté, c'est un objet mathématique abstrait,
  - de l'autre, c'est un code qui va finir par être exécuté sur du matériel.
- ▶ Le premier point de vue correspond à celui de la cryptanalyse classique.
- ▶ Le second correspond à celui de la sécurité physique.
- ▶ Les attaques physiques tirent partie des caractéristiques spécifiques des implémentations pour retrouver les paramètres secrets utilisés pendant le calcul.
- ▶ Ces attaques sont donc moins générales que celles de la cryptanalyse classique, mais elles sont aussi beaucoup plus puissantes.

- ▶ Il existe de nombreux types d'attaques physiques.
- ▶ À haut niveau, on peut déjà les classer selon deux axes :
  - *invasives* ou *non-invasives* : faut-il ouvrir ou casser en partie l'implémentation, ou au contraire n'exploiter que des informations naturellement (bien que non-intentionnellement) émises ?
  - *active* ou *passive* : l'attaque agit-elle sur l'implémentation ou se contente-t-elle de l'observer ?

- ▶ Il existe de nombreux types d'attaques physiques.
- ▶ À haut niveau, on peut déjà les classer selon deux axes :
  - *invasives* ou *non-invasives* : faut-il ouvrir ou casser en partie l'implémentation, ou au contraire n'exploiter que des informations naturellement (bien que non-intentionnellement) émises ?
  - *active* ou *passive* : l'attaque agit-elle sur l'implémentation ou se contente-t-elle de l'observer ?
- ▶ Aujourd'hui, on va s'intéresser aux attaques non-invasives passives.

- ▶ Les attaques passives exploitent des grandeurs physiques observables durant l'exécution du code, qui dépendent des données sensibles.
- ▶ Ces grandeurs peuvent être
  - le temps,
  - la consommation de courant,
  - des émanations électromagnétiques,
  - la chaleur,
  - le bruit,
  - ...

- ▶ Si on ne fait pas attention, le temps d'exécution de la plupart des algorithmes dépend des données.
- ▶ Par exemple, pour l'exponentiation modulaire  $B^E \bmod M$  :

---

```
1 r := 1
2 b := b % m
3 tant que e ≠ 0:
4     si e & 1 = 1 alors:
5         r := (r * b) % m
6     fin
7     b := (b * b) % m
8     e := e >> 1
9 fin
```

---

- ▶ Ce genre d'attaque peut fonctionner à travers le réseau.

- ▶ La contre-mesure est évidente : rendre l'exécution des programmes constante en temps.
- ▶ C'est par défaut le cas de la plupart des algorithmes de chiffrement symétrique par exemple.
- ▶ Mais c'est parfois plus subtil qu'il n'y paraît :
  - accès mémoire,
  - cache des processeurs.
- ▶ De fait, on sait depuis 2005 que AES peut-être vulnérable à ce genre d'attaque.



- ▶ Ces attaques sont très puissantes.
- ▶ La raison de cela est qu'on arrive très bien à modéliser la consommation de courant, et qu'elle est fortement corrélée aux données.
- ▶ En pratique, les mesures de consommations sont très proportionnelles
  - au poids de Hamming des valeurs (nombre de bits à 1),
  - à la distance de Hamming entre les valeurs qui se succèdent dans un même bus ou registre (nombre de bitflips).
- ▶ Les attaques par analyse d'émanations électromagnétiques correspondent à la même chose (c'est en fait l'activité électrique qui produit ces émanations).

# Attaques

- ▶ Il existe différentes formes d'attaques.
- ▶ Les deux principales sont
  - la *SPA*, pour *simple power analysis*, et
  - la *DPA*, pour *differential power analysis*.

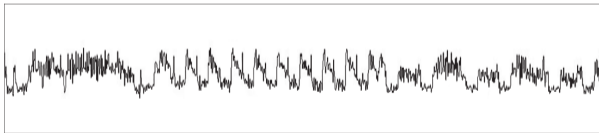
# Monter une attaque

- ▶ Monter une attaque par analyse de consommation de courant nécessite plusieurs appareils.
- ▶ Au minimum, il est nécessaire d'avoir :
  - la cible (smartcard, FPGA, etc.) et une "board" permettant de l'utiliser.
  - une sonde (antenne EM, résistance, etc.),
  - un outil d'acquisition de trace de consommation (par exemple un oscilloscope qui peut prendre au moins 1Gsample/s, et sensible au  $\mu\text{A}$ ), et
  - un ordinateur pour analyser les traces de consommation.
- ▶ Avec autant d'appareil, le gros défi est de minimiser le bruit et/ou de réussir à isoler l'information pertinente dans les traces.

# Analyse simple de consommation

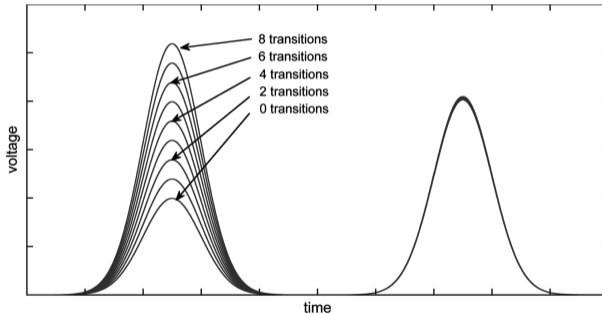
- ▶ Le principe d'une SPA est de ne regarder qu'une seule trace de consommation, correspondant à une seule exécution du système cible.
- ▶ Cela peut déjà donner énormément d'informations :

- ▶ Le principe d'une SPA est de ne regarder qu'une seule trace de consommation, correspondant à une seule exécution du système cible.
- ▶ Cela peut déjà donner énormément d'informations :
  - par exemple sur le même code que pour l'attaque temporelle, on voit les activités différentes à chaque tour en fonction de la valeur des bits de l'exposant (1 vs 2 activités, voire 2 activités différentes si le `square` est optimisé),
  - on peut aussi identifier l'algorithme (par exemple en voyant le nombre de tours),
- ▶ Exemple de trace de consommation lors de l'exécution d'un AES 128 :



## Analyse différentielle de consommation

- ▶ Le principe de la DPA est d'utiliser plusieurs (parfois des centaines de milliers !) traces de consommation.
- ▶ Des outils statistiques permettent alors d'exploiter les dépendances entre les données.
- ▶ Typiquement, avec les consommations moyennes toujours à un même point dans le temps on peut évaluer le nombre de bitflips à cette étape du calcul en utilisant le modèle de fuite dont on a déjà parlé (la distance de Hamming).



- ▶ Il est aussi possible de déjouer certaines contre-mesures qui rajoutent du bruit grâce à certaines méthodes statistiques.

## Autres techniques

- ▶ Il existe bien d'autres techniques d'analyse de consommation, dont certaines encore plus poussées :
  - attaque par *templates*,
  - ASCA (pour *Algebraic Side-Channel Attack*),
  - CPA (pour *Correlation Power Analysis*),
  - ...
  
- ▶ Nous ne les détailleront pas dans ce cours, mais vous pouvez choisir un article à synthétiser sur ces sujets là !

- ▶ Contre ce type d'attaques, il existe des contre-mesures aux niveaux matériel et logiciel.
- ▶ Les plus efficaces sont bien sûr celles qui mêlent les deux : des contre-mesures logicielles supportées par du matériel spécialisé.
- ▶ En tant qu'informaticien-nes, nous allons nous intéresser principalement aux contre-mesures logicielles.
- ▶ Il y a deux familles de contre-mesures :
  - les contre-mesures *palliatives*, et
  - les contre-mesures *curatives*.



# Contre-mesures palliatives

- ▶ Les contre-mesures palliatives tentent d'utiliser de l'aléatoire pour ajouter du bruit dans les fuites d'information et les rendre inexploitable.
- ▶ Cependant, cela est fait sans vrai fondement théorique et en dehors d'un cadre formel.

- ▶ Les contre-mesures curatives tentent de faire disparaître complètement l'information pertinente dans les fuites pour les rendre inexploitable.
- ▶ Pour cela elles s'appuient sur une formalisation des objectifs de sécurité.
- ▶ Il existe essentiellement deux telles stratégies :
  - le *masquage* consiste à rendre la fuite aussi décorrélée que possible des données sensibles,
  - l'*équilibrage* consiste à rendre la fuite constante, c'est à dire indépendante des données sensibles.

- ▶ L'idée de masquage et de mixer les données sensibles avec des nombres aléatoires pendant qu'on les manipule, de façon à en décorréler la fuite.
- ▶ Les avantages de cette techniques sont :
  - son indépendance vis-à-vis du matériel,
  - l'existence de systèmes prouvés.
- ▶ Cependant elle a aussi des inconvénients :
  - la possibilité d'attaques assez puissantes pour passer outre le masquage,
  - sa forte demande de nombres aléatoires, très coûteux à générer et pouvant se révéler être une faille supplémentaire.

- ▶ Le but de l'équilibrage est de rendre la fuite constante, c'est à dire de la rendre totalement indépendante des données sensibles.
- ▶ Pour cela, il y a besoin d'une collaboration de la part du matériel : il faut que celui-ci fournisse deux ressources qui soient indistinguables du point de vue des canaux auxiliaires, c'est à dire qu'elles doivent fuir de manière identique.
- ▶ Cela peut paraître étonnant mais ça n'est pas du tout évident !
- ▶ Ces deux ressources vont être utilisées dans ce qu'on appelle un protocole *double rail*.

- ▶ La contre-mesure DPL (*Dual-rail with Precharge Logic*) consiste à faire tous les calculs sur une représentation redondante : chaque bit  $y$  est représenté par une paire  $(y_{\text{False}}, y_{\text{True}})$ .
  - ▶ La paire de bits est utilisées en respectant un protocole en deux phases :
    - la *précharge*, lors de laquelle les deux bits de la paires sont remis à zéro :  $(y_{\text{False}}, y_{\text{True}}) = (0, 0)$  ;
    - l'*évaluation*, lors de laquelle la paire  $(y_{\text{False}}, y_{\text{True}})$  est mise à  $(1, 0)$  si le bit logique  $y$  vaut 0 ou à  $(0, 1)$  si le bit logique  $y$  vaut 1.
- Étudions la mise en œuvre de cette contre-mesure au niveau logiciel [RGN14].