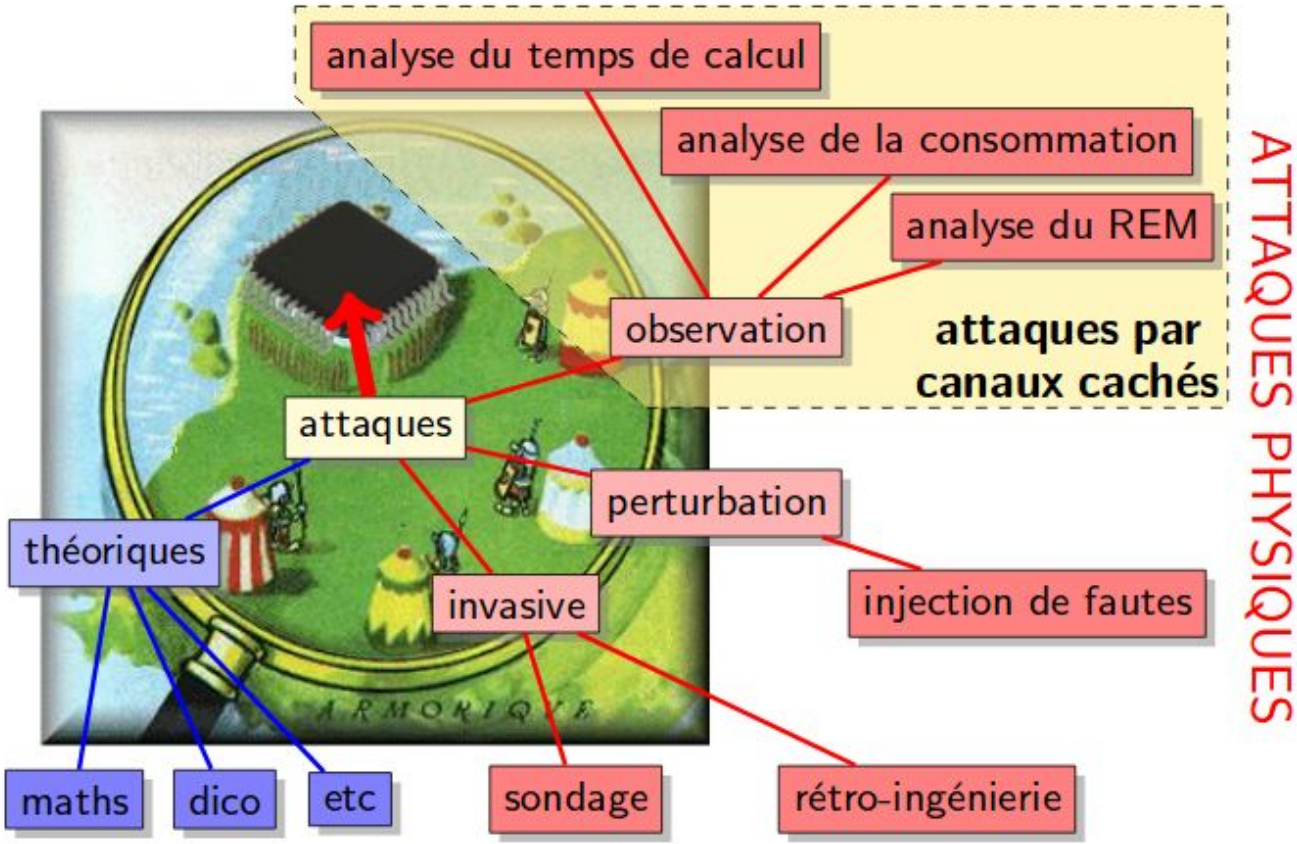
Les attaques physiques, qu’est ce que c’est ?

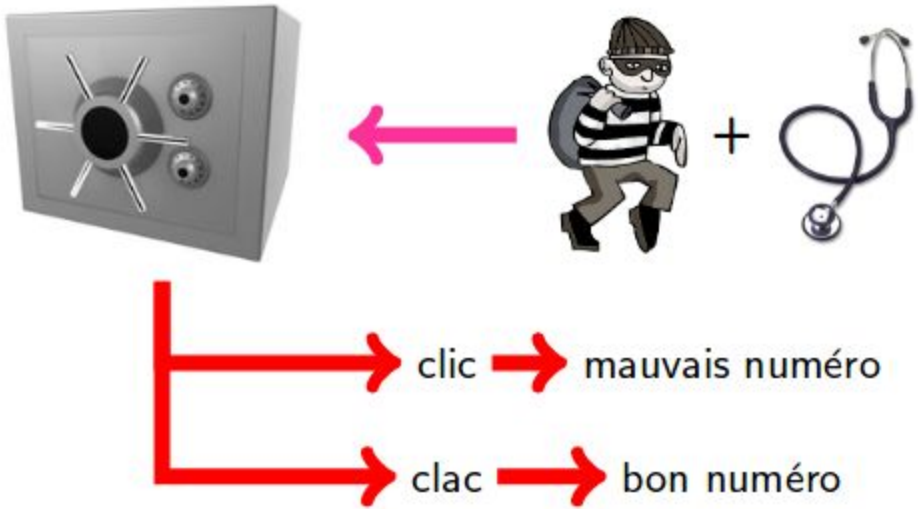
Principaux types d'attaques

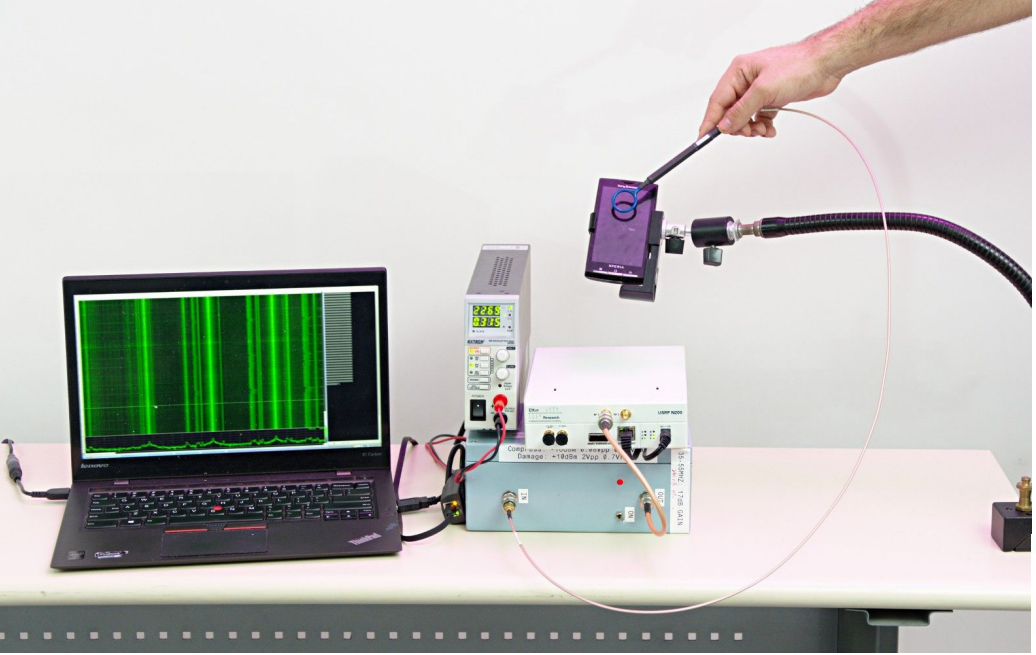


REM = rayonnement électromagnétique

Attaques par canaux cachés, une nouvelle technique ?

Pas vraiment ! Vous connaissez tous des attaques par canaux cachés...





Researchers use Side-Channel attack to steals Encryption keys from Android and iOS Devices

Side-channel attack has been used earlier to hack into air gapped computers but this is the first time researchers have used this vector to steal encryption keys from Android smartphones and iOS devices. Five researchers from universities in Tel Aviv and Adelaide have devised a new crypto side-channel attack that can extract encryption keys from electromagnetic emanations coming out of Android and iOS devices that are running cryptographic operations.

For the uninitiated, in cryptography, a side-channel attack is any attack based on information gained from the physical implementation of a cryptosystem, rather than brute force or theoretical weaknesses in the algorithms (compare cryptanalysis). For example, timing information, power consumption, electromagnetic leaks or even sound can provide an extra source of information, which can be exploited to break the system.

Quelles grandeurs physiques mesurer ?

Réponse : tout ce qui « entre » ou « sort » dans le ou du circuit

- temps de calcul
- consommation d'énergie
- rayonnement électromagnétique (REM)
- température
- bruit
- nombre de défauts de cache d'un ordinateur
- nombre et type des messages d'erreur
- ...

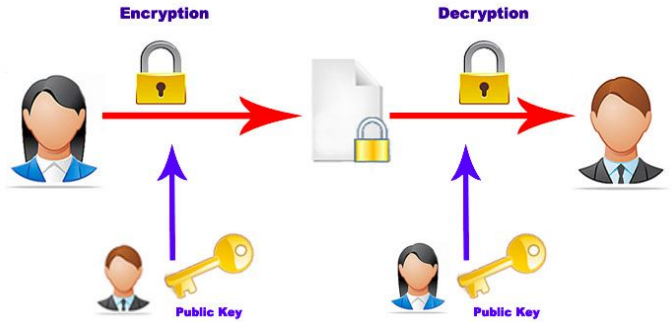
Les valeurs mesurées peuvent révéler des informations sur :

- le comportement global (conso., REM, température, bruit ...)
- le comportement local (REM local, nb. déf. cache ...)

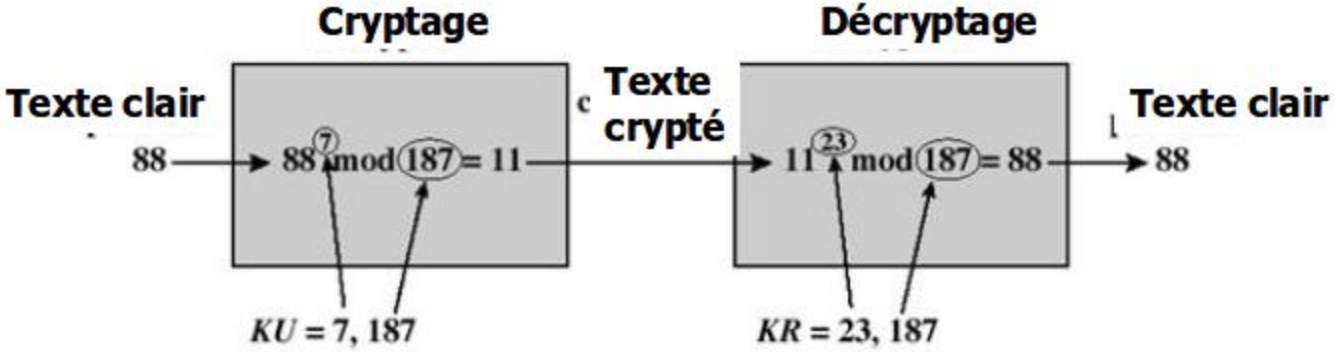
Simple Power Analysis (SPA)

- Interprétation directe de la consommation du circuit
- Vise les opérations effectuées et la clef!
- Trace: courbe temporelle du courant
- Une opération à 1ms échantillonnée à 5MHz donne une trace avec 5000 points

RSA et chiffrement asymétrique : Un algorithme très répandu !



- Étapes
  1. Sélectionner deux entiers premiers entre eux « p » et « q »
  2. Calculer n = p x q
  3. Calculer φ(n)=(p-1)(q-1)
  4. Sélectionner « e » tel que: *pgcd*(φ(n),e)=1 ; 1<e<φ(n)
     - *En général « e » est un entier de petite taille.*
  5. Calculer $d=e^{-1}$ mod φ(n). En d'autre terme: d.e = 1 mod (φ(n))
  6. Clé publique: K_{pu}={e,n}
  7. Clé privée K_{pr} = {d,n}
- Pour crypter un message M < n, l'émetteur:
  - Obtient une clé publique du récepteur et calcule « $C= M^e$ mod n »
- Pour décrypter un message crypté C le récepteur
  - Utilise sa clé privée et calcule « $M = C^d$ mod n »



- p = 17, q = 11, n = p x q= 187
- Φ(n) = 16 x 10 =160,
- Choisir e = 7,
- d.e =1 (mod Φ(n)) ➔ d = 23

Faille de RSA : l'exponentiation

La méthode la plus simple pour faire l'exponentation d'un entier par un autre est la méthode d'exponentiation rapide.

(basé sur la décomposition binaire de la clé K)

```
Input:    X, N, K=(k_{j-1}, ..., k_1, k_0)_2
Output:   Z = X^K mod N

1:    Z = 1;
2:    for i=j-1 downto 0 {
3:        Z = Z * Z mod N //Square
4:        if (k_i==1) Z = Z * X mod N //Multiply
5:        }
6:    return(Z);
```

Faille de RSA : l'exponentiation

La méthode la plus simple pour faire l'exponentation d'un entier par un autre est la méthode d'exponentiation rapide.
Sa consommation électrique est directement liée à l'exposant.

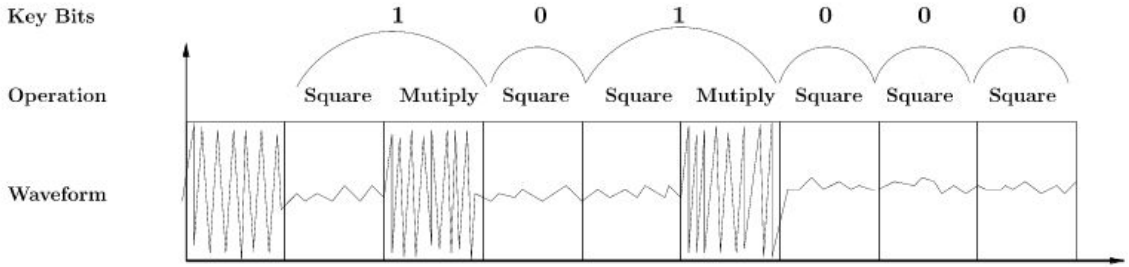
Simple Power Analysis (SPA)

Cette méthode consiste à retrouver des informations par l'analyse de la courbe représentative de la consommation de courant lors de l'exponentiation à la puissance d.
Cette courbe de consommation est différente suivant les instructions exécutées et les données manipulées.

SPA sur RSA

```
Input:    X, N, K=(k_{j-1}, …, k_1, k_0)_2
Output:   Z = X^K mod N

1:    Z = 1;
2:    for i=j-1 downto 0 {
3:        Z = Z * Z mod N //Square
4:        if (k_i==1) Z = Z * X mod N //Multiply
5:        }
6:    return(Z);
```



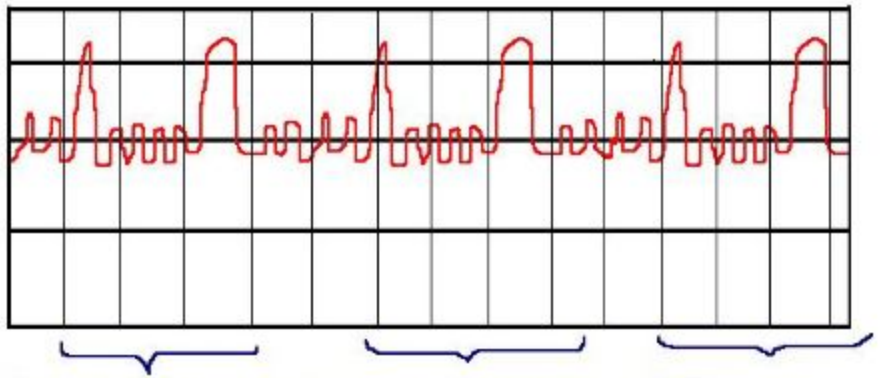
Contre-mesure sur RSA

- Faire en sorte que la consommation soit constante. Introduire des opérations factices.

Input: X, N, $K=(k_{j-1}, \ldots, k_1, k_0)_2$
Output: $Z = X^K \bmod N$

```
1:      Z = 1;
2:      for i=j-1 downto 0 {
3:              Z = Z * Z mod N //Square
4:              if (k_i==1) Z = Z * X mod N //Multiply
5:              else  U = Z * X mod N
6 :     }
6:      return(Z);
```

Courbe de consommation de l'algorithme transformé



Toujours la même courbe de consommation quelque soit le bit en cours

Conclusion

L'utilisation des fuites d'informations permet de retrouver des informations sur le secret utilisé.
L'ajout d'opérations factices empêche de lire directement la clé.
Pour retrouver malgré tout des informations :
Essayer de déterminer quelles sont les opérations factices, ce qui peut se faire en provoquant des erreurs.

"C'est en faisant des erreurs que l'on apprend."

Attaques par fautes

Principe

- Les attaques par canaux cachés utilisent les failles matérielles de l'implantation pour récupérer des informations sur le secret utilisé.
- Les attaques par injection de fautes consistent à perturber l'exécution d'un algorithme par exemple par des émissions lasers.
- L'analyse du résultat nous permettra de déduire de l'information sur le secret.

$\Rightarrow$ Voyons comment nous pouvons appliquer cette attaque.

Attaque par injection de faute

Cible de l'attaque

```
Input:      X, N, K=(k_{j-1}, ..., k_1, k_0)_2
Output:     Z = X^K mod N

1:      Z = 1;
2:      for i=j-1 downto 0 {
3:              Z = Z * Z mod N //Square
4:              if (k_i==1) Z = Z * X mod N //Multiply
5:              else  U = Z * X mod N
6 :     }
6:      return(Z);
```

que par rapport à l'algorithme d'exponentiation classique, nous sommes passés de n élévations au carrés et (en moyenne) $n/2$ multiplications, à exactement n élévations au carrés et n multiplications, si n est la taille du module N.

Attaques par fautes

Schéma d'attaque

Cible de l'attaque

Nous allons entrainer des perturbations durant l'exécution de l'algorithme au moment où les pas d'addition sont effectués. En

- provoquant des sur tensions
- utilisant un laser
- ou en chauffant l'appareil...

Conséquences

De deux choses l'une :

- soit le résultat final est faussé, et donc l'addition est bien effectuée on sait que le bit visé est à 1.
- soit le résultat final n' est pas faussé, et donc l'addition est factice on sait que le bit visé est à 0.

Attaque par injection de faute

Résultat de l'attaque

- L'attaque par faute consiste ainsi à perturber l'exécution d'un algorithme pour en déduire de l'information sur le secret utilisé.

- Nous venons de voir comment une implémentation protégée contre les attaques SPA peut être tout de même attaquée par une attaque par faute.

- Nous voulons maintenant protéger l'implémentation de l'exponentiation contre ces deux attaques.
- Il nous faut construire un algorithme d'exponentiation rapide dont la trace de courant soit régulière, sans qu'aucune opération ne soit factice.

$\Rightarrow$ Any idea ?

Attaque par injection de faute

Contre mesure à cette attaque

Utilisation de deux variables => calculs symétriques dans les cas 0 et 1!

La méthode de l'échelle de Montgomery (Montgomery ladder)

Algorithme d'exponentiation rapide

On veut calculer C^d

1 - Calculer la décomposition binaire de d, $d = \overline{d_n d_{n-1} \dots d_1 d_0}^2$
2 - $T \leftarrow C$ $U \leftarrow C^2$
3 - **Si** $d_i = 0$ **Alors** $T \leftarrow T^2$ et $U \leftarrow T \times U$
4 - **Si** $d_i = 1$ **Alors** $T \leftarrow T \times U$ et $U \leftarrow T^2$
5 - **Renvoyer** T

Attaque par injection de faute

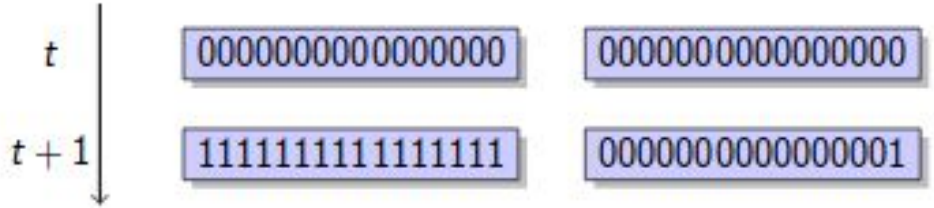
Aller plus loin ?

- Nous avons introduit les attaques SPA.
- Elle permettent d'attaquer l'exponentiation rapide.
- Nous avons construit des contre mesures à cette attaque.

- Nous avons de nouveau fait une attaque : attaques par faute.
- Nous venons de protéger l'exponentiation contre cette attaque.
- Pouvons nous de nouveau attaquer ?...

Limites de la SPA

Exemple de différence de comportement : (activité dans un registre)



Important : une petite variation de comportement peut être plus ou moins cachée par du bruit $\Longrightarrow$ les traces ne sont plus discernables

Question : que faire quand les différences sont (trop) petites ?

Réponse : utiliser des statistiques sur des nombreuses courbes

Differential Power Analysis (DPA)

- Utilisation de méthodes statistiques pour trouver de petites variations qui peuvent être masqués par le bruit ou des erreurs de mesures
- Exploite les informations fuyant de l'implantation physique du cryptosystème

DPA principe

Plus puissante et plus difficile à prévenir que la SPA

Consommation de puissance différente suivant les états (0 ou 1), ou transitions (0->1 et 1->0)

1/ Phase d'acquisition de courbes (clair aléatoire)

2/ Procédure

- Hypothèse sur une clef
- Diviser les courbes en 2 groupes (A et B pour un bit choisi)
- Calculer la moyenne des courbes de chaque groupe
- Hypothèse incorrecte : moyennes égales
- Hypothèse correcte → différence non négligeable

DPA principe

- On applique des données connues et on mémorise la courbe de consommation (globale)
- On vise un bit particulier (par exemple un bit de sortie de l'AES)
- On corrèle la valeur du bit au fait qu'il y a consommation sur ce bit ou pas selon une hypothèse sur la clef
- On extrait par des méthodes statistiques la consommation liée à ce bit de la conso. globale.
- On vérifie si il y a une corrélation.
- Si la corrélation est bonne, hypothèse sur la clef OK
- Sinon, hypothèse KO

DPA Procedure pour l'AES

1. Faire la mesure de conso de 1000 opérations AES, 100000 pts / courbe, (Ciphertext$_i$, Courbe$_i$)
2. Hypothèse sur une clef associée à une S-box à la dernière ronde
   1. Calculer le premier bit d'entrée de la S-box pour chaque chiffré en se basant sur l'hypothèse de clef
   2. Diviser l'ensemble des courbes en 2 groupes PA et PB (entrée 0 et entrée 1)

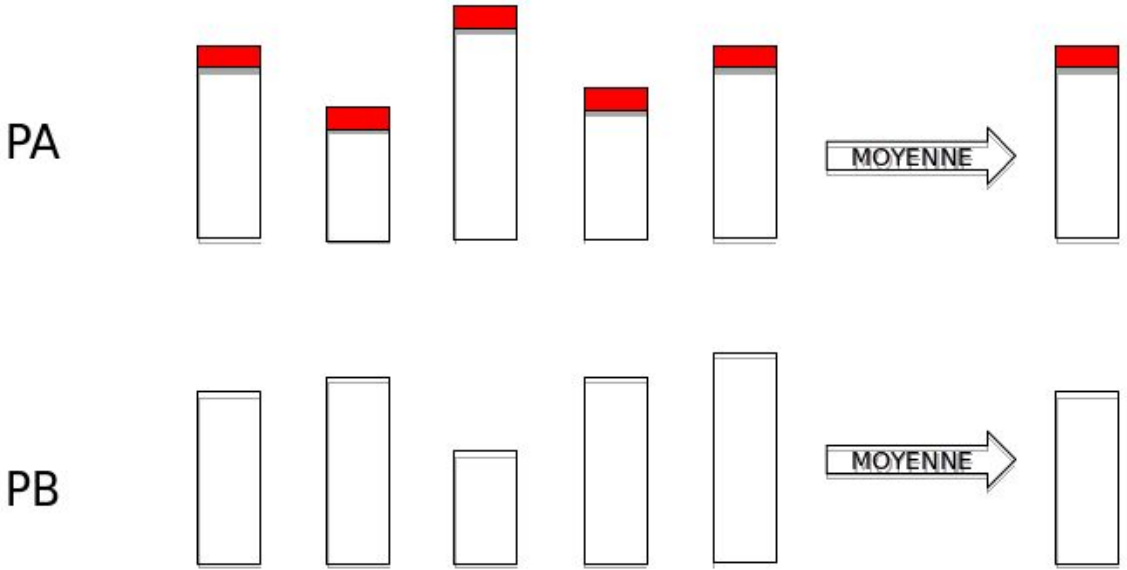
      PA = contient les courbes où le bit "consomme de la puissance"

      PB = contient les courbes où le bit "ne consomme pas de la puissance"

S-box = chiffrement d'un group de 8 bits

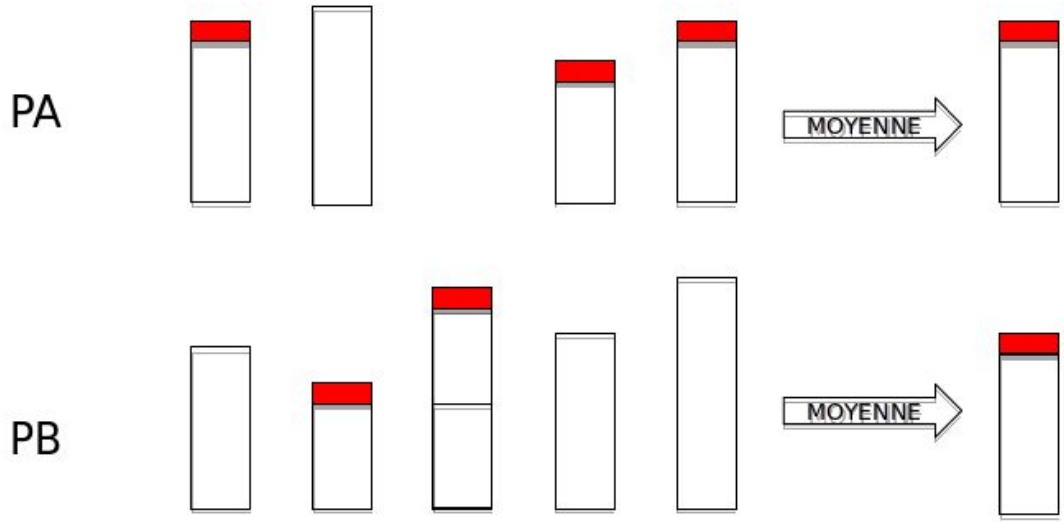
3. Calculer la courbe moyenne de chaque groupe
4. Calculer la différence des 2 courbes
5. Si l'hypothèse est correcte → pics sur la courbe de différences
6. Répéter 2-5 pour les autres bits et autres S-boxes

Pour le bonne hypothèse de clef : PA contient toujours une composante de la puissance qui n'est pas dans PB.



 Conso(PA) > Conso(PB)

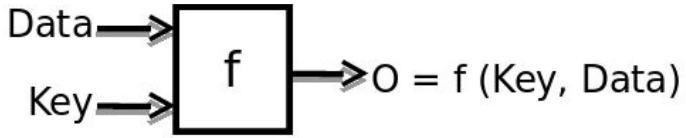
Pour une mauvaise hypothèse de clef : la consommation liée à ce bit se répartit entre PA et PB



 Conso(PA) ≈ Conso(PB)

DPA: Faire l'attaque

- Noeud cible:
  - un bit dont la valeur dépend du message d'entrée et d'une petite partie de la clef (e.g. 8 bits)
- Idées de base:
  - On essaie toutes les clefs (si 8 bits, 256 hypothèses)
  - Pour chaque hypothèse de clef, création de 2 ensembles, création de deux ensembles PA et PB
  - Sélection de la valeur de clef correspondant à la plus grande valeur “Power(PA) – Power(PB)”



Inputs: $T_1, T_2, \ldots, T_{12}$

Key: K_x (8 bits)

Contre-mesures DPA

- Masquage des données par une valeur aléatoire
- Techno duale (doubler les bus, implanter f et f')
- Désynchronisation (introduire des temps d'attente aléatoires, décalage des courbes)
- Ajout de bruit
- etc.....

Conclusion

Nous venons d'introduire les attaques par canaux cachés, dont le principe est d'utiliser des failles de l'implémentation pour récupérer de l'information sur le secret utilisé. Nous avons décrit les attaques :

- SPA,
- attaque par fautes
- DPA

Il en existe d'autre par exemple attaque par temps, ou differentielle en faute.