TELECOM ParisTech

INSTITUT Mines-Télécom

Side-Channel Analysis

Attaques par canaux auxiliaires

Guillaume Duc, Ulrich Kühne
ulrich.kuhne@telecom-paristech.fr
2018–2019

Outlines

Objectifs du cours

- Comprendre la notion d'attaque par canal auxiliaire
- Comprendre le fonctionnement des attaques SCA (*Side-Channel Analysis*) classiques
- Connaître les différentes catégories de contre-mesures et comprendre leur fonctionnement

Contexte général

- Algorithme
- Implémenté
  - Directement en matériel (ASIC, FPGA...)
  - Ou en logiciel s'exécutant sur un processeur (soft-core dans un FPGA, processeur généraliste dans un système embarqué ou un PC, processeur spécialisé...)
- Remplissant un objectif de sécurité
  - Confidentialité (exemple : algorithme de chiffrement)
  - Authentification (exemple : verification d'un code PIN)
  - ...
- En manipulant un secret (qui peut être l'algorithme lui-même) qui ne doit par être accessible par l'adversaire

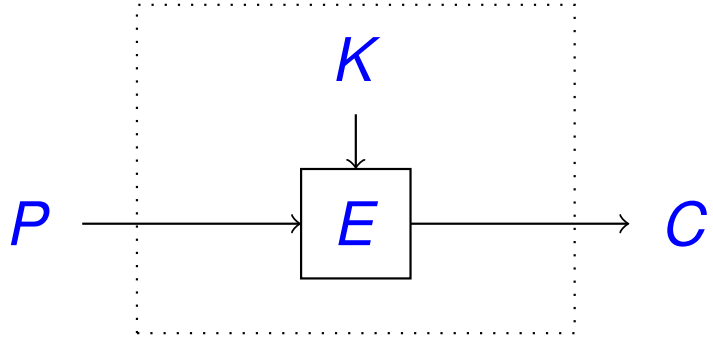
Outlines

Outlines

Exemple

- Exemple : Algorithme de chiffrement implémenté sur une carte à puce
- Entrée : message à chiffrer
- Sortie : message chiffré
- Par conception, la clé de chiffrement est embarquée dans la carte et n'est pas accessible via des commandes sur l'interface entrée/sortie de la carte

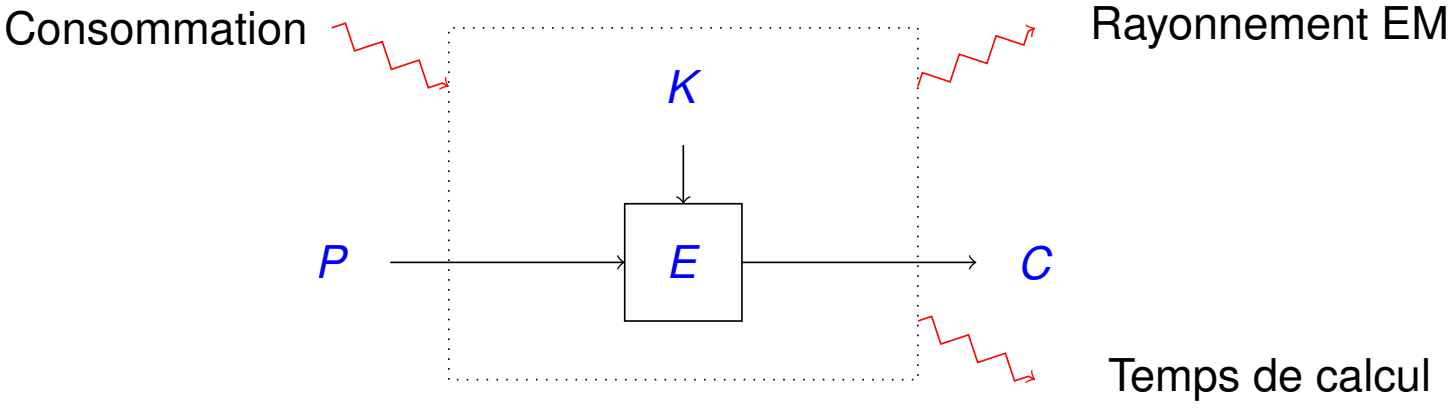
Vue mathématique



- Principes de KERCKHOFFS : P, C et E sont publics et la sécurité ne dépend que de K qui est inconnue de l'adversaire
- De nombreux algorithmes robustes en considérant ce modèle

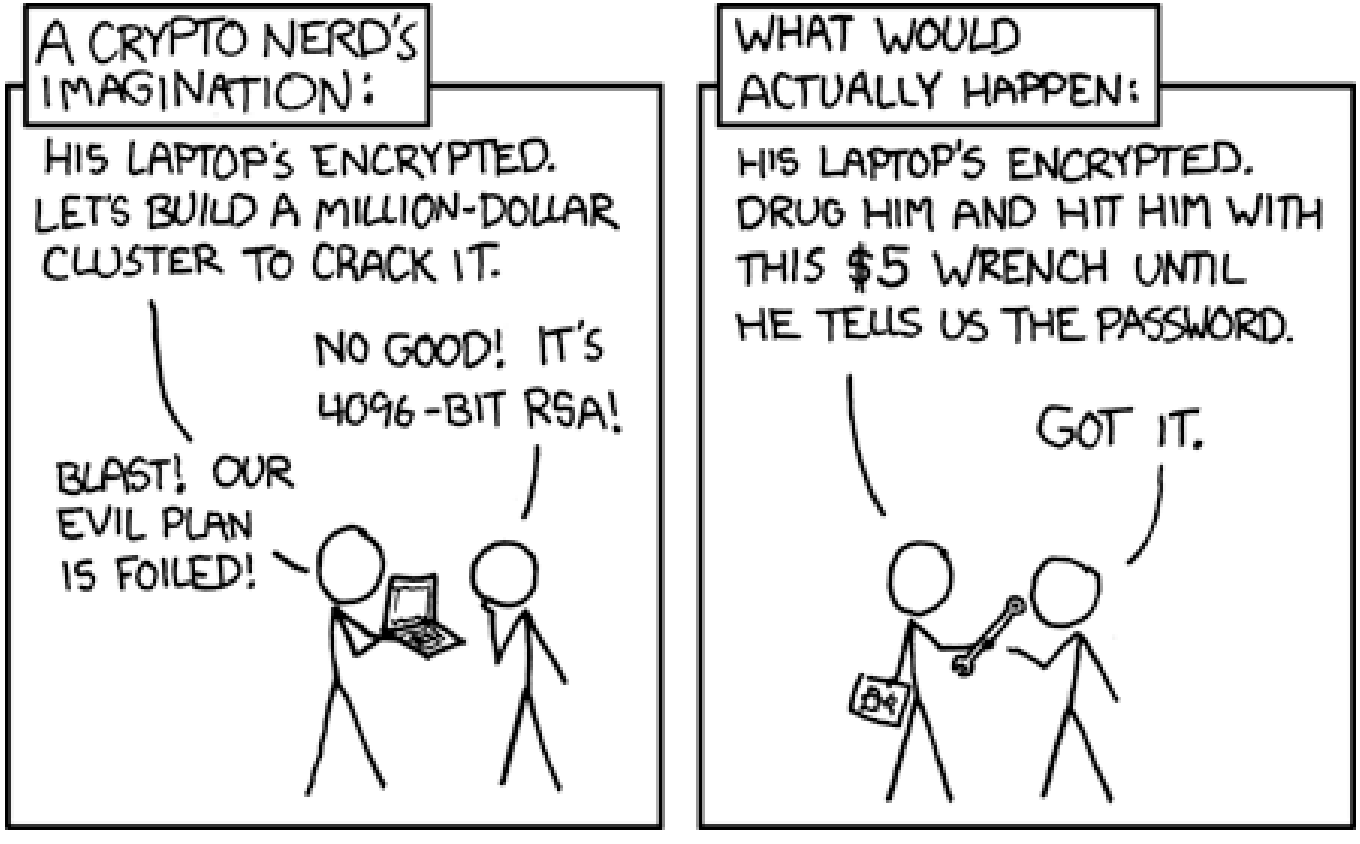
Dans la vraie vie...

... le matériel



- Canaux d'entrées/sorties supplémentaires : Canaux auxiliaires (*side-channels*)
  - Rayonnement EM
  - Consommation électrique
  - Temps de calcul
  - ...

Cryptanalysis vs Reality...



[Source : https://www.xkcd.com/538/]

Attaque par canal auxiliaire

- Ces canaux auxiliaires sont liés à l'implémentation de l'algorithme (que ce soit sous forme matérielle ou logicielle), ils n'apparaissent pas si on étudie simplement l'algorithme mathématiquement
- L'implémentation de l'algorithme peut laisser fuir des informations sensibles (secret) via ces canaux auxiliaires
- Donc bien qu'aucune information sensible ne soit accessible sur les canaux de communication normaux de l'algorithme, une écoute passive des canaux auxiliaires introduits par son implémentation peut permettre à l'adversaire de découvrir tout ou partie des secrets

Exemple concret

Fonction de vérification d'un code PIN

```
boolean verifyPIN(byte[] inputPIN)
{
  for (int i = 0; i < correctPIN.length; i++)
    if (inputPIN[i] != correctPIN[i])
      return false;

  return true;
}
```

- On supposera que les tableaux `inputPIN` et `correctPIN` sont de taille 4 et contiennent uniquement des chiffres (0–9)
- Quelle est la complexité de l'attaque exhaustive (essayer tous les codes PIN) ?
- L'attaquant peut-il être plus rusé ?

Exemple concret

Fonction de vérification d'un code PIN

- Il suffit pour l'attaquant de mesurer le temps de réponse de l'algorithme
- En effet, la fonction de vérification s'arrête dès qu'elle rencontre un chiffre erroné
- Donc l'attaquant essaie 0xxx, 1xxx, 2xxx, ..., 9xxx et pour l'un d'entre eux, le temps de réponse sera un peu plus long, indiquant ainsi quel est le premier chiffre correct
- Il peut ensuite répéter le test pour le deuxième chiffre, etc.
- Complexité de l'attaque : maximum 40 tests contre 9999 tests pour l'attaque exhaustive
- Le canal auxiliaire utilisé ici par l'adversaire est le temps de calcul $\rightsquigarrow$ *timing attack*

Outlines

Outlines

Consommation d'un circuit CMOS

- Rappel : Si on néglige les courants de fuite (consommation statique), un circuit CMOS ne consomme que lors des changements d'état de ses portes (consommation dynamique)
- Donc en observant la consommation d'un circuit, on peut en déduire son activité
- Or, le nombre de portes logiques qui changent d'état est lié aux opérations effectuées ainsi qu'aux données manipulées
- Donc la consommation peut révéler des informations sur les opérations effectuées et les données manipulées, y compris des données devant rester secrètes

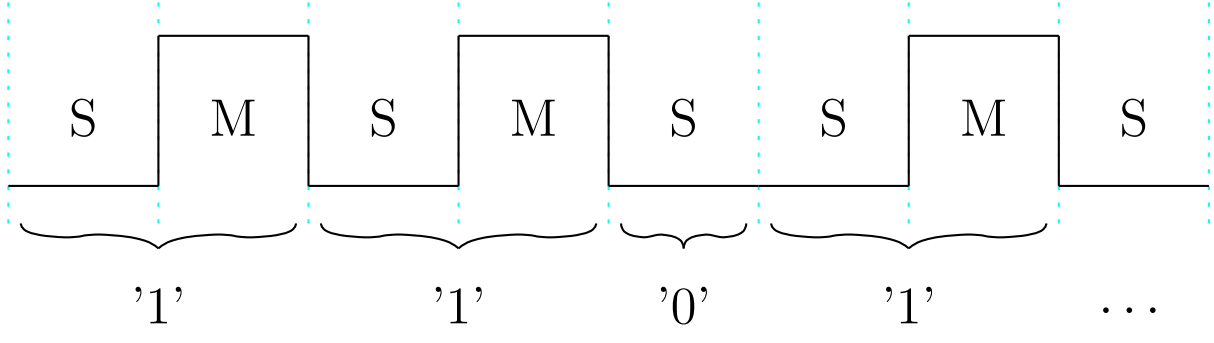
Simple Power Analysis (SPA)

Exemple de RSA

- Algorithme de calcul d'une exponentiation modulaire

Inputs : M , K
$R = 1$;
for $i = |K| - 1$; $i \geq 0$; $i - -$ **do**
  $R = R^2$;
  if $K_i == 1$ **then**
    $R = R \times M$;
  end if
end for
Return $R = M^K$;

- Profil de consommation



Simple Power Analysis (SPA)

Exemple de RSA

- En effectuant une seule mesure de consommation du circuit, il est possible de retrouver complètement le secret (dans le cas de RSA, la clé secrète)
- Cette attaque est possible car l'algorithme n'effectue pas les mêmes opérations en fonction du secret (ici, en fonction de chaque bit du secret) et que ces opérations (ici multiplication et mise au carré) ont un profil de consommation différent
- Il s'agit d'une SPA (*Simple Power Analysis*) car elle ne nécessite qu'une seule mesure
- On peut noter que l'algorithme est également vulnérable à une attaque par mesure du temps de calcul (moins intéressante ici)...

Outlines

DPA

- Souvent, la fuite n'est pas si flagrante et il est nécessaire de collecter un grand nombre de mesures (de consommation ou de rayonnement EM) et d'utiliser des outils statistiques pour extraire l'information voulue (*c'est presque du Big Data...*)
- Il s'agit des attaques DPA (*Differential Power Analysis*) et de ses dérivées (CPA...)

Outils nécessaires

Modèle de fuite Modèle prédisant le comportement du système sur le canal auxiliaire considéré en fonction d'une hypothèse sur son état

Distingueur Outil statistique permettant de détecter la présence d'une corrélation entre le comportement réel du circuit et celui prédit par le modèle de fuite

- Le modèle de fuite (ou modèle de consommation) prédit la consommation du circuit en fonction de son état (opérations effectuées, données manipulées, etc.)
- Comme les opérations effectuées ou les données manipulées ne sont pas toutes connues par l'adversaire (notamment la clé), ce modèle dépend d'une hypothèse sur les éléments inconnus (hypothèse qui peut être vraie ou fausse)

Mode opératoire 1/2

- Identifier une variable sensible S qui dépend d'une partie du secret (pas la totalité sinon l'attaque revient à une attaque exhaustive) et des entrées ou sorties connues
- Établir un modèle de fuite dépendant de S : $M(S)$
- Effectuer des observations (mesures) du comportement du circuit sur le canal auxiliaire considéré en fonction de différentes entrées et/ou sorties

Mode opératoire 2/2

- Pour chaque valeur possible de S
  - Pour chaque entrées/sorties utilisées pour effectuer les observations, calculer $M(S)$
  - Utiliser le distingueur pour vérifier s'il existe une corrélation entre le comportement prédit par le modèle (et qui dépend de l'hypothèse) et les observations réelles
- Pour la valeur correcte de S, le modèle de fuite prédit correctement le comportement du circuit et donc les observations vont être corrélées au modèle et donc le distingueur va repérer cette corrélation
- Pour toutes les autres valeurs de S, le modèle de fuite ne prédit pas correctement le comportement du circuit et donc les observations ne vont pas être corrélées au modèle

Réalisation de l'attaque

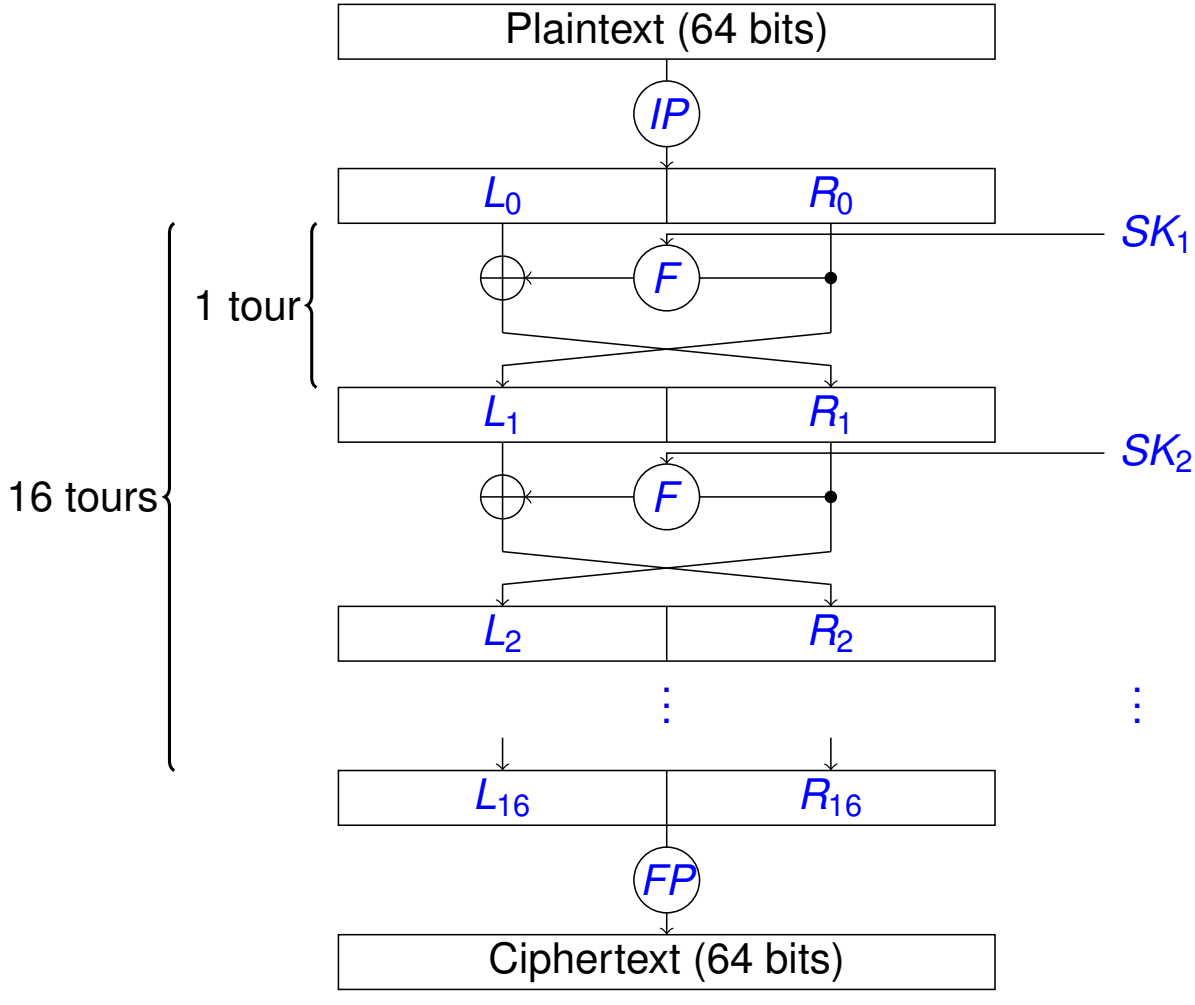
1. Quel modèle de fuite choisir ?
2. Quel distingueur choisir ?
3. Comment effectuer les observations ?

Exemple

- Contexte : implémentation matérielle itérative de l'algorithme DES (opération de chiffrement en mode ECB)
- Information à retrouver : clé (56 bits)
- L'adversaire peut envoyer des messages en clair au circuit, enregistrer les messages chiffrés résultants et mesurer en temps réel la consommation du circuit pendant l'opération de chiffrement
- Méthode utilisée : DPA (*Differential Power Analysis*)

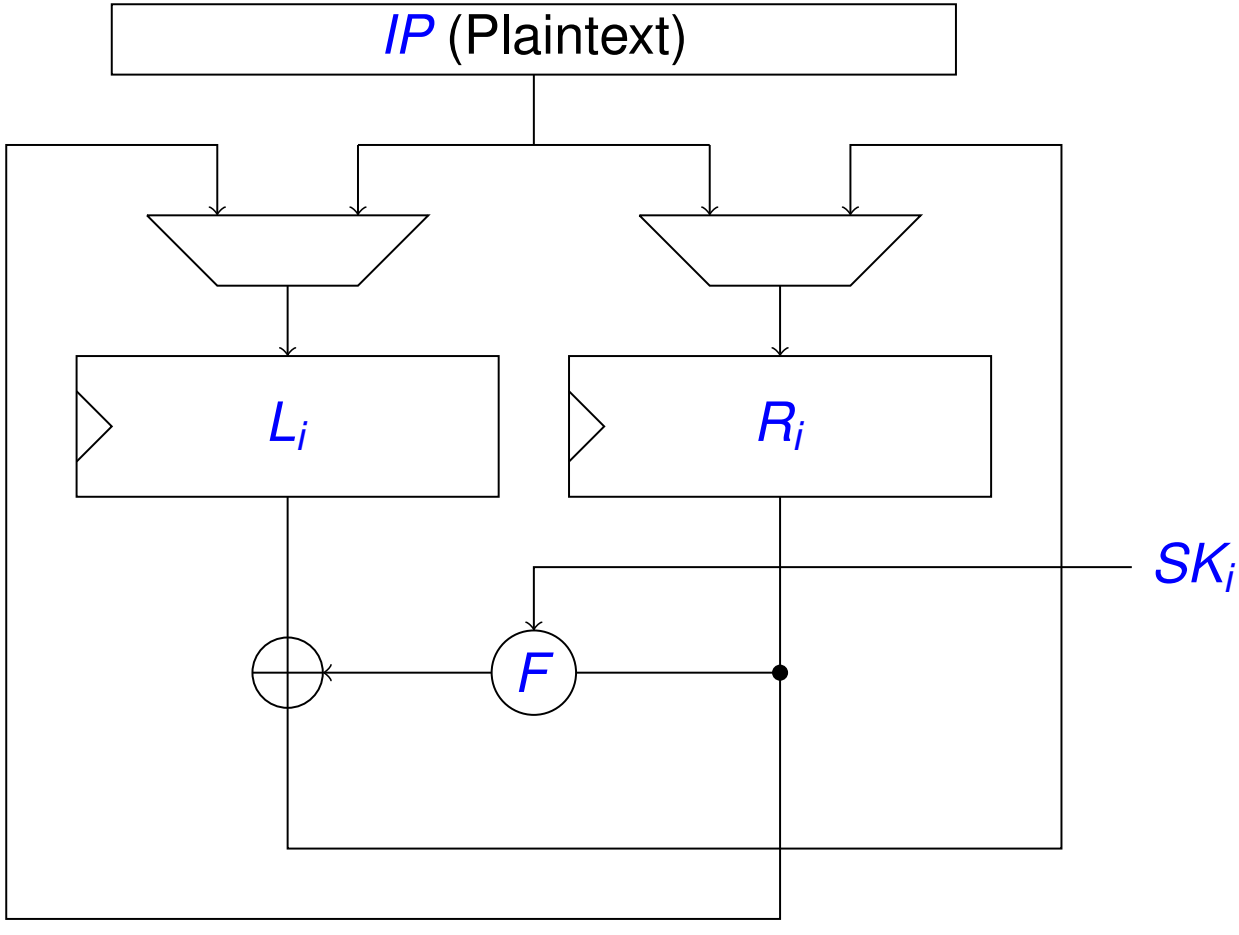
Exemple : DPA sur DES

DES : vue algorithmique



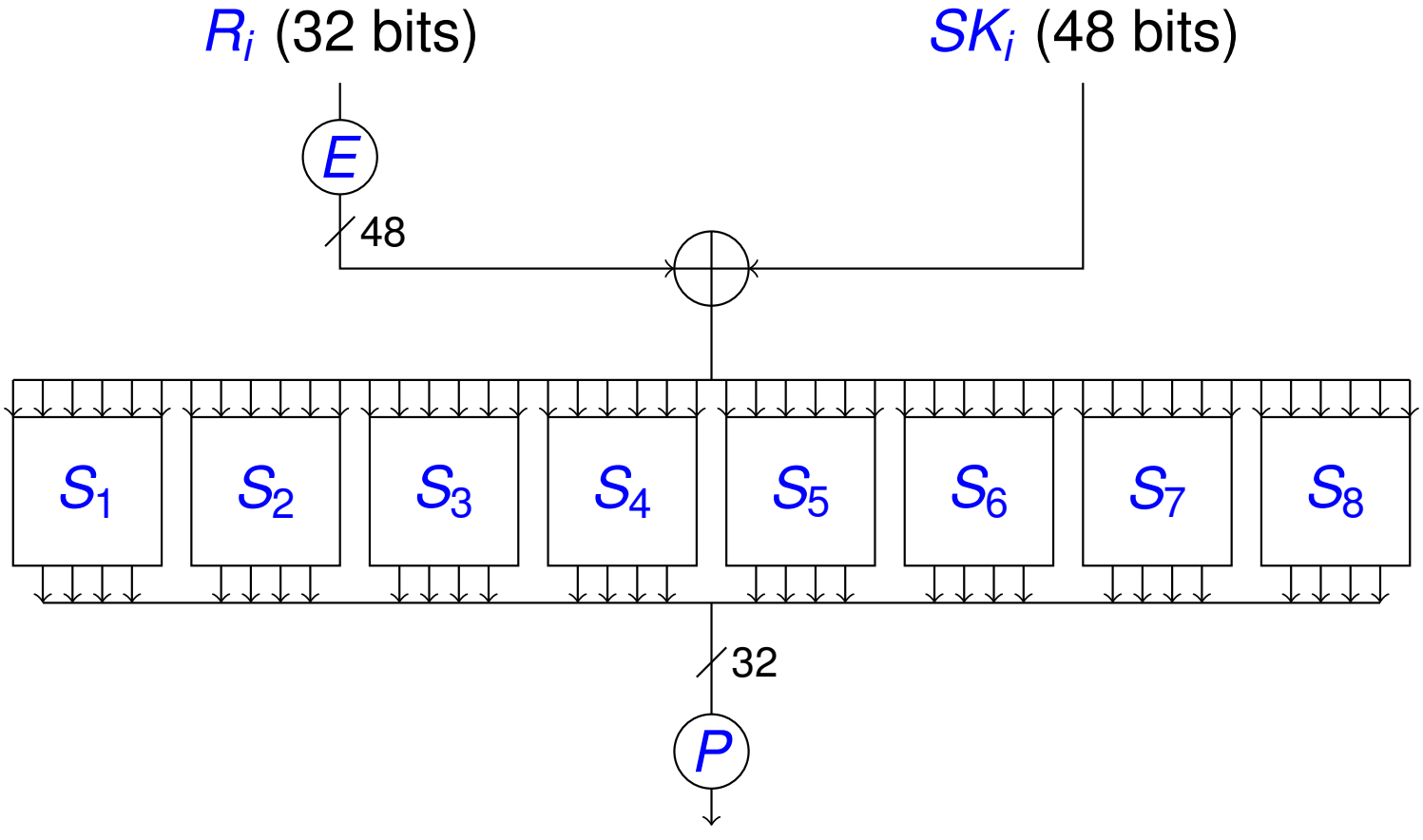
Exemple : DPA sur DES

DES : implémentation matérielle itérative



Exemple : DPA sur DES

Fonction de Feistel (DES)



Exemple : DPA sur DES

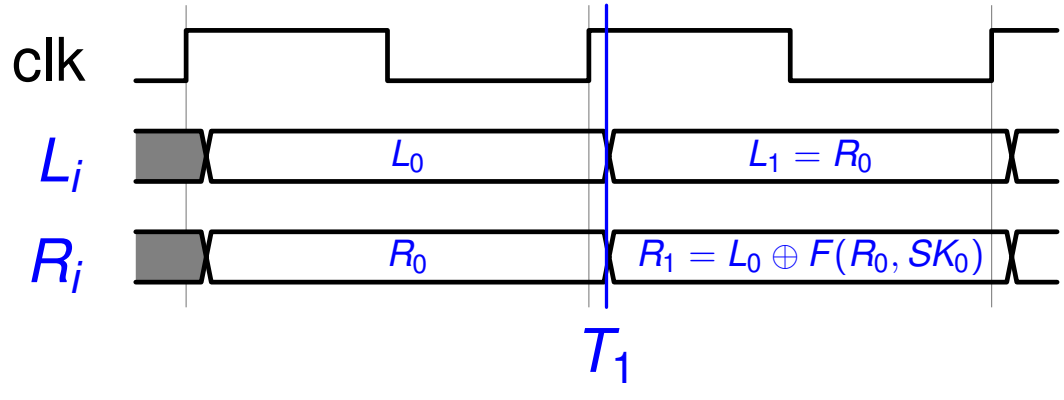
Modèle de consommation

- Quel modèle de consommation choisir ?
- Consommation du circuit pendant l'opération de chiffrement
- Problèmes
  - DES n'est pas seul sur la puce (entrées/sorties...)
  - La consommation de la partie DES dépend fortement de la clé (56 bits) or il n'est pas possible de tester toutes ces hypothèses (revient à faire une attaque exhaustive)
- Il va falloir se concentrer sur la consommation d'une partie seulement du circuit dépendant simplement d'une fraction de la clé et considérer que la consommation du reste du circuit est du bruit

Exemple : DPA sur DES

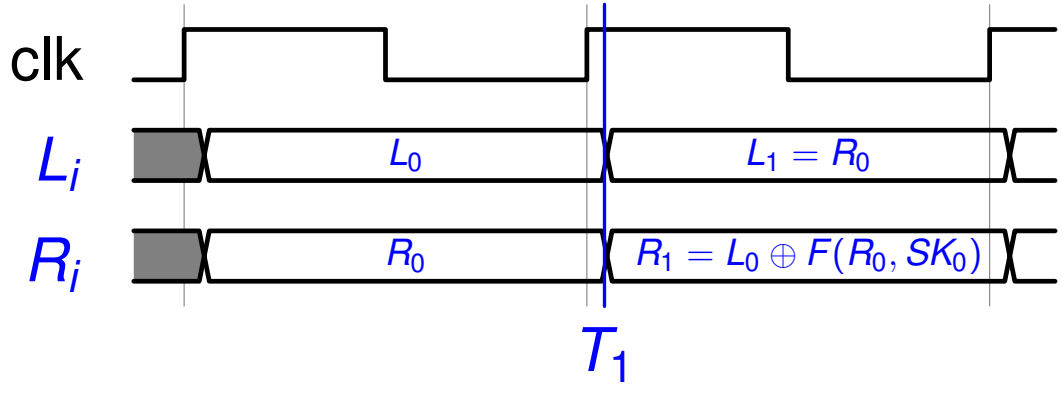
Registres d'état du chemin de données

- Évolution du contenu des registres d'état (L_i et R_i) du chemin de données au cours d'une opération de chiffrement



Exemple : DPA sur DES

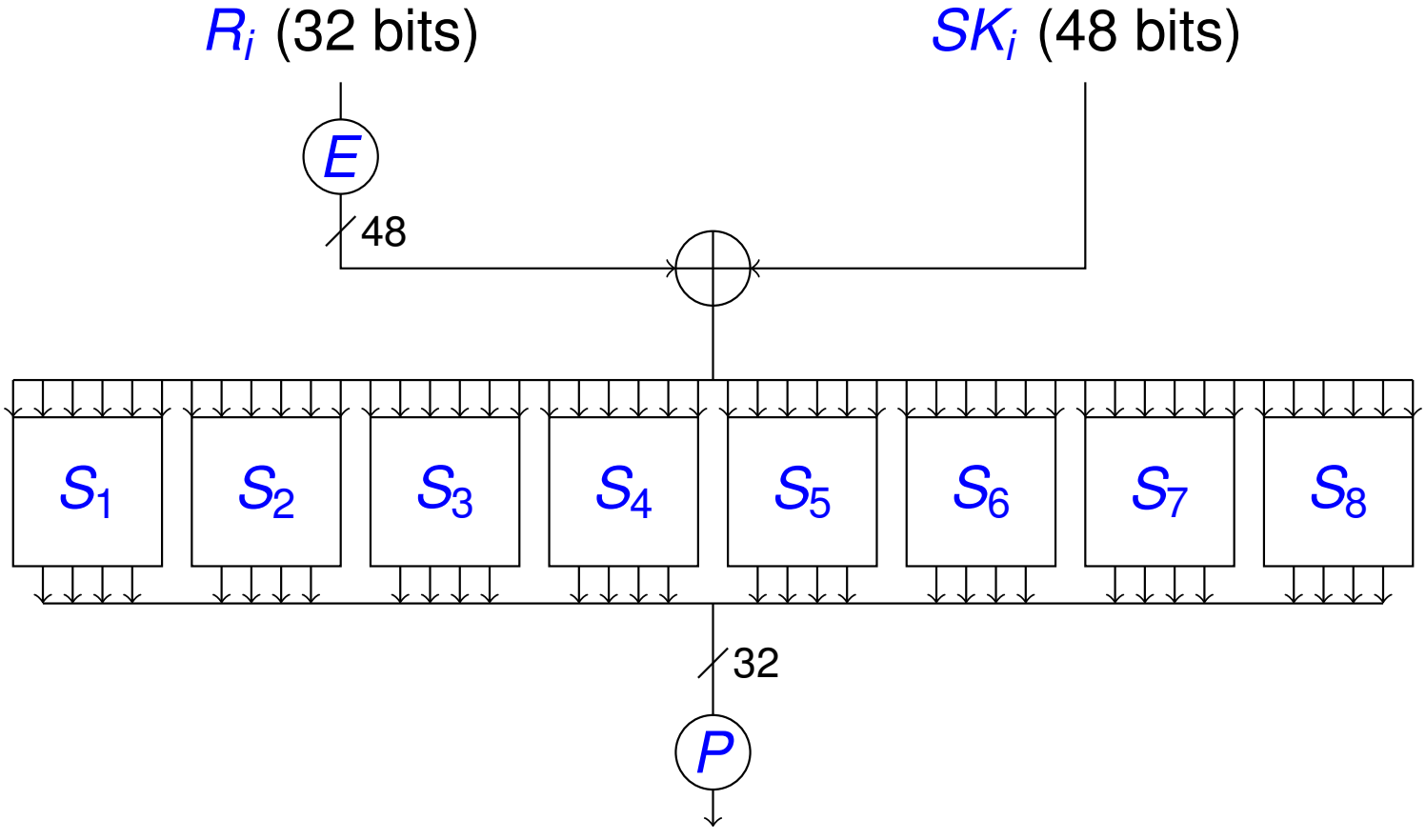
Consommation des registres d'état



- Consommation du registre R_i à l'instant T_1 : $C_{R_i}(T_1) = \delta \times \mathrm{HD}(R_0, L_0 \oplus F(R_0, SK_0))$
- Variables connues : R_0 et L_0 (dépendent directement du message en clair)
- Variables inconnues : SK_0 (48 bits provenant de la clé K), T_1 et δ
- Encore trop d'hypothèses à faire (2^{48})

Exemple : DPA sur DES

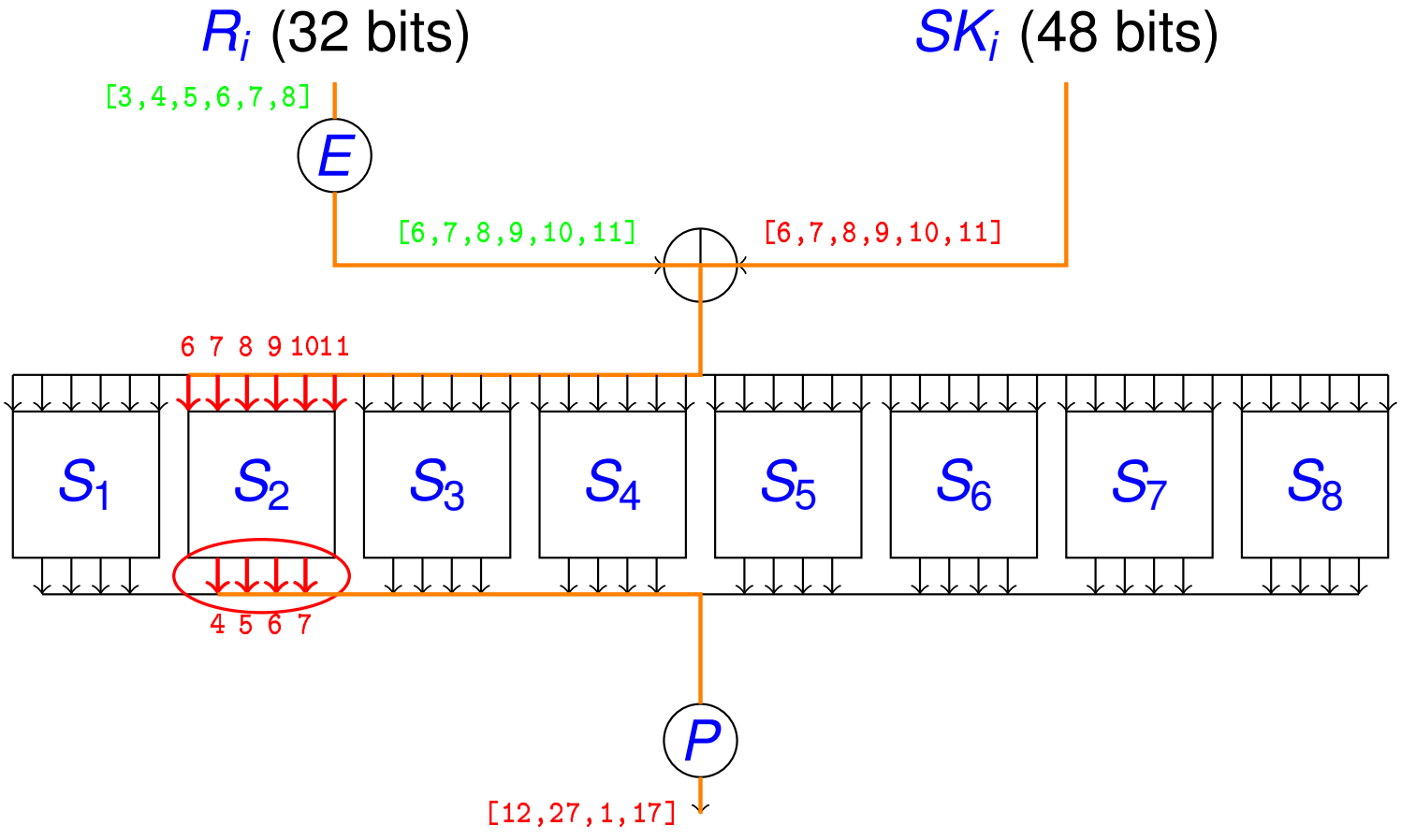
Zoom sur la fonction de Feistel



- Comment établir un modèle de consommation dépendant de moins de bits du secret ?

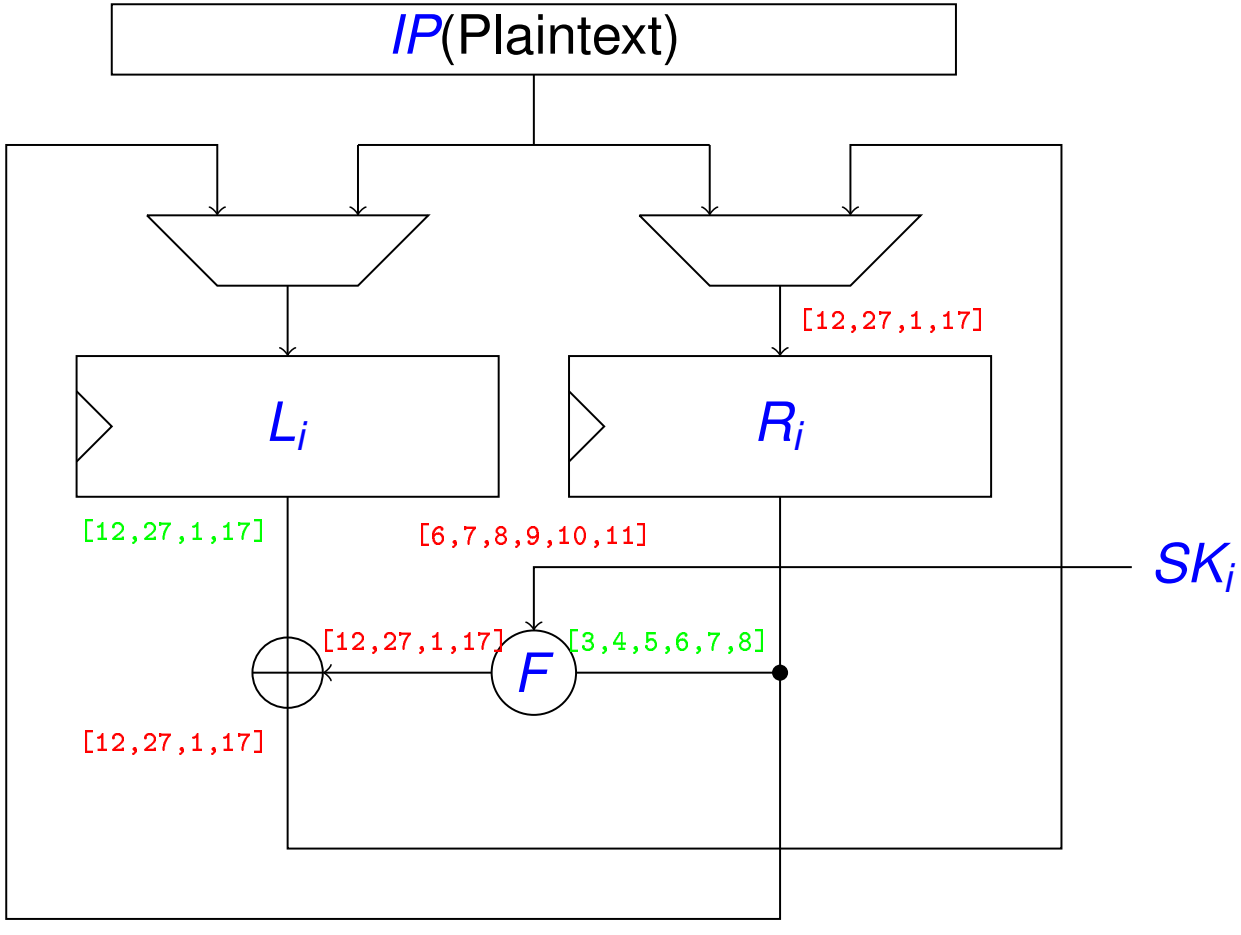
Exemple : DPA sur DES

Zoom sur la fonction de Feistel (premier tour)



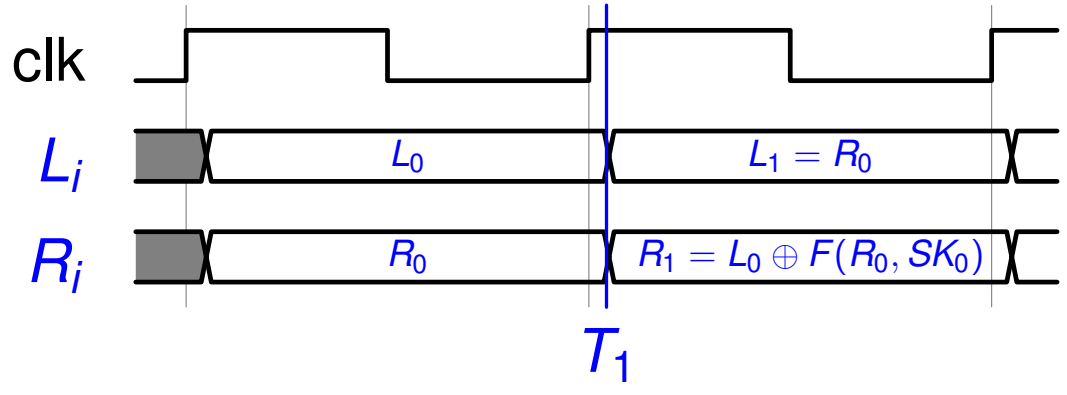
Exemple : DPA sur DES

Impact de la SBox 2 (premier tour)



Exemple : DPA sur DES

Consommation des registres d'état (impact SBox 2)



- On s'intéresse aux bits `[12,27,1,17]` du registre R_i
- Avant T_1, leur valeur dépend de R_0 et donc directement du message en clair (connu)
- Après T_1, leur valeur dépend
  - Des bits `[12,27,1,17]` de L_0 (connus)
  - Des bits `[3,4,5,6,7,8]` de R_0 (connus)
  - Des bits `[6,7,8,9,10,11]` de SK_0 (inconnus)

Exemple : DPA sur DES

Modèle de consommation HD sur 4 bits

- Modèle de consommation : $C_{R_i[12,27,1,17]}(T_1) = \delta \times \mathrm{HD}(R_0[12,27,1,17], L_0[12,27,1,17] \oplus F(R_0[3,4,5,6,7,8], SK_0[6,7,8,9,10,11]))$
- Dépend d'une hypothèse sur 6 bits de la sous-clé numéro 0 ($2^6 = 64$ hypothèses possibles)
- Ce modèle n'est valide qu'à l'instant T_1
- 5 valeurs possibles (distance de hamming sur 4 bits) : $0, \delta, 2\delta, 3\delta, 4\delta$
- Dans la suite, on suppose que $\delta = 1$
- Finalement : $C_4(P, S) = C_{R_i[12,27,1,17]}(T_1)$ où
  - P est le message en clair
  - S est l'hypothèse sur $SK_0[6,7,8,9,10,11]$

Exemple : DPA sur DES

Modèle de consommation vs. consommation réelle

- Le modèle de consommation ne prédit que la consommation d'une petite partie du circuit (4 bascules D) et uniquement à un instant précis (T_1)
- Consommation réelle à l'instant T_1 :

$$C_{réelle}(P, K, T_1) = C_4(P, S_T) + C_{reste}(P, K, T_1)$$

où S_T correspond à la bonne hypothèse (bonne valeur de $SK_0[6, 7, 8, 9, 10, 11]$ en fonction de K)

- On va supposer que $C_{reste}(P, K, T_1)$ est statistiquement indépendant de $C_4(P, S_T)$

Exemple : DPA sur DES

Acquisitions

- Pour la bonne hypothèse sur S (S_T), à l'instant T_1, la consommation réelle va dépendre en partie du modèle de consommation ($C_4(P, S)$)
- Cette dépendance étant faible, il va falloir beaucoup de mesures pour la faire ressortir grâce au distingueur
- On effectue donc des mesures de la consommation du circuit pour de nombreux messages en clair différents (la clé restant la même) : $P_0, \ldots, P_{N-1}$ (où N est le nombre de mesures effectuées)

Exemple : DPA sur DES

Acquisitions

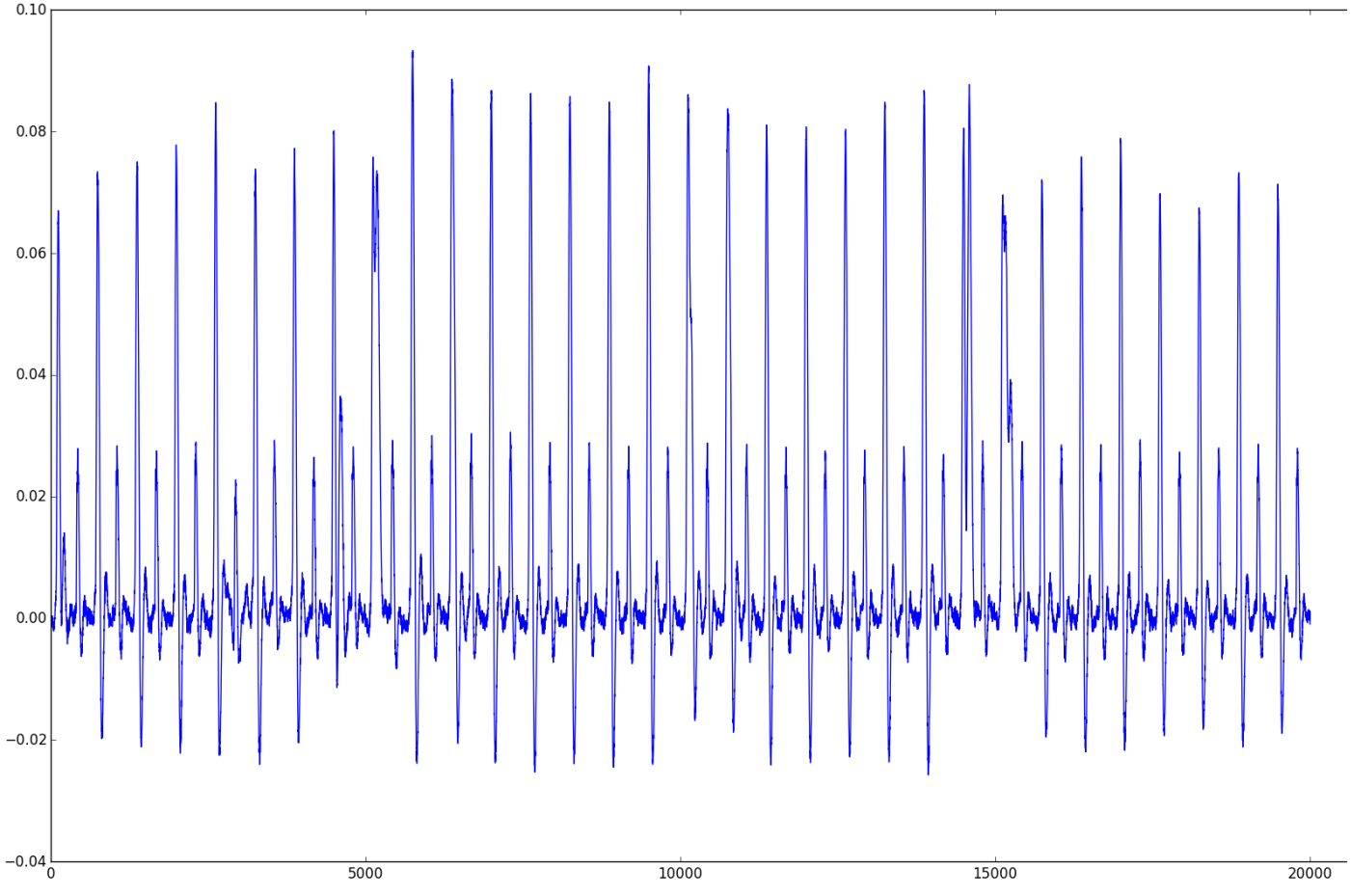
- Mesure de la consommation du circuit pendant un chiffrement = trace de consommation
- Trace = vecteur d'échantillons : $C_{mesurée}(P_i, K, t)$ pour $t = t_0, \ldots, t_{M-1}$ (où M est le nombre d'échantillons par trace)

$$C_{mesurée}(P_i, K, t) = C_{réelle}(P_i, K, t) + Bruit_{Acquisition}$$

- On suppose par la suite que toutes les traces sont synchronisées, c'est-à-dire que le numéro de l'échantillon correspondant à l'instant T_1 est le même pour toutes les traces

Exemple : DPA sur DES

Exemple de trace de consommation



- Unités arbitraires (x : temps, y : consommation)

Exemple : DPA sur DES

Mode opératoire de l'attaque

1. On fait une hypothèse sur $S = S_H$ (64 valeurs possibles, parmi elle, la valeur correcte : S_T)
2. On partitionne l'ensemble des traces en fonction de la prédiction du modèle de consommation : pour chaque trace $C_{mesurée}(P_i, K, t)$ ($i = 0, \ldots, N-1$)
   - On calcule la prédiction du modèle de consommation : $C_4(P_i, S_H)$ (5 valeurs possibles)
   - On classe la trace dans un des 5 ensembles $E_{C_4=0}, \ldots, E_{C_4=4}$:

$$E_{C_4=j} = \{C_{mesurée}(P_i, K, t) \mid C_4(P_i, S_H) = j\}$$

Exemple : DPA sur DES

Mode opératoire de l'attaque

3. Dans chacun des 5 ensembles, on calcule une trace moyenne (chaque échantillon i de cette trace est la moyenne arithmétique des échantillons i de toutes les traces de cet ensemble) :

$$C_{moy_{C_4=j}}(t) = \frac{1}{n} \sum_{C_{mesurée} \in E_{C_4=j}} C_{mesurée}(P_i, K, t)$$

pour $t = 0, \ldots, M-1$ et où n est le nombre de traces dans l'ensemble $E_{C_4=j}$

Exemple : DPA sur DES

Mode opératoire de l'attaque

4. On calcule enfin une trace différentielle :

$$C_{diff}(t) = -2 \times C_{moy_{C_4=0}}(t) - C_{moy_{C_4=1}}(t) + C_{moy_{C_4=3}}(t) + 2 \times C_{moy_{C_4=4}}(t)$$

   pour $t = 0, \ldots, M-1$
5. On cherche ensuite l'échantillon maximum de la trace différentielle : $D(S_H) = \max_t C_{diff}(t)$
6. On cherche enfin, pour quelle hypothèse sur S, $D(S_H)$ est maximal. Normalement, cette hypothèse est la bonne (S_T)

Exemple : DPA sur DES

Pourquoi l'attaque fonctionne ?

- On a :

$$C_{mesurée}(P_i, K, t) = C_{réelle}(P_i, K, t) + Bruit_{Acquisition}$$

- À l'instant T_1 :

$$C_{réelle}(P, K, T_1) = C_4(P, S_T) + C_{reste}(P, K, T_1)$$

- D'où :

$$C_{mesurée}(P_i, K, T_1) = C_4(P_i, S_T) + C_{reste}(P_i, K, T_1) + Bruit_{Acquisition}$$

- On considère le bruit d'acquisition et la consommation du reste du circuit comme étant du bruit B :

$$C_{mesurée}(P_i, K, T_1) = C_4(P_i, S_T) + B$$

Exemple : DPA sur DES

Pourquoi l'attaque fonctionne ? (hypothèse correcte)

- On suppose tout d'abord que l'on fait l'hypothèse correcte sur S ($S_H = S_T$)
- Lorsque l'on applique le modèle de consommation, ce dernier prédit correctement, dans tous les cas, le comportement des 4 bits étudiés du registre R_i
- De ce fait, le partitionnement effectué sur l'ensemble des traces est cohérent par rapport au comportement réel de ces 4 bits : pour $j = 0, \ldots, 4$, $\forall C_{mesurée}(P_i, K, t) \in E_{C_4=j}$, on a :

$$C_{mesurée}(P_i, K, T_1) = j + B$$

Exemple : DPA sur DES

Pourquoi l'attaque fonctionne ? (hypothèse correcte)

- Lors du calcul des traces moyennes, cette cohérence est conservée :

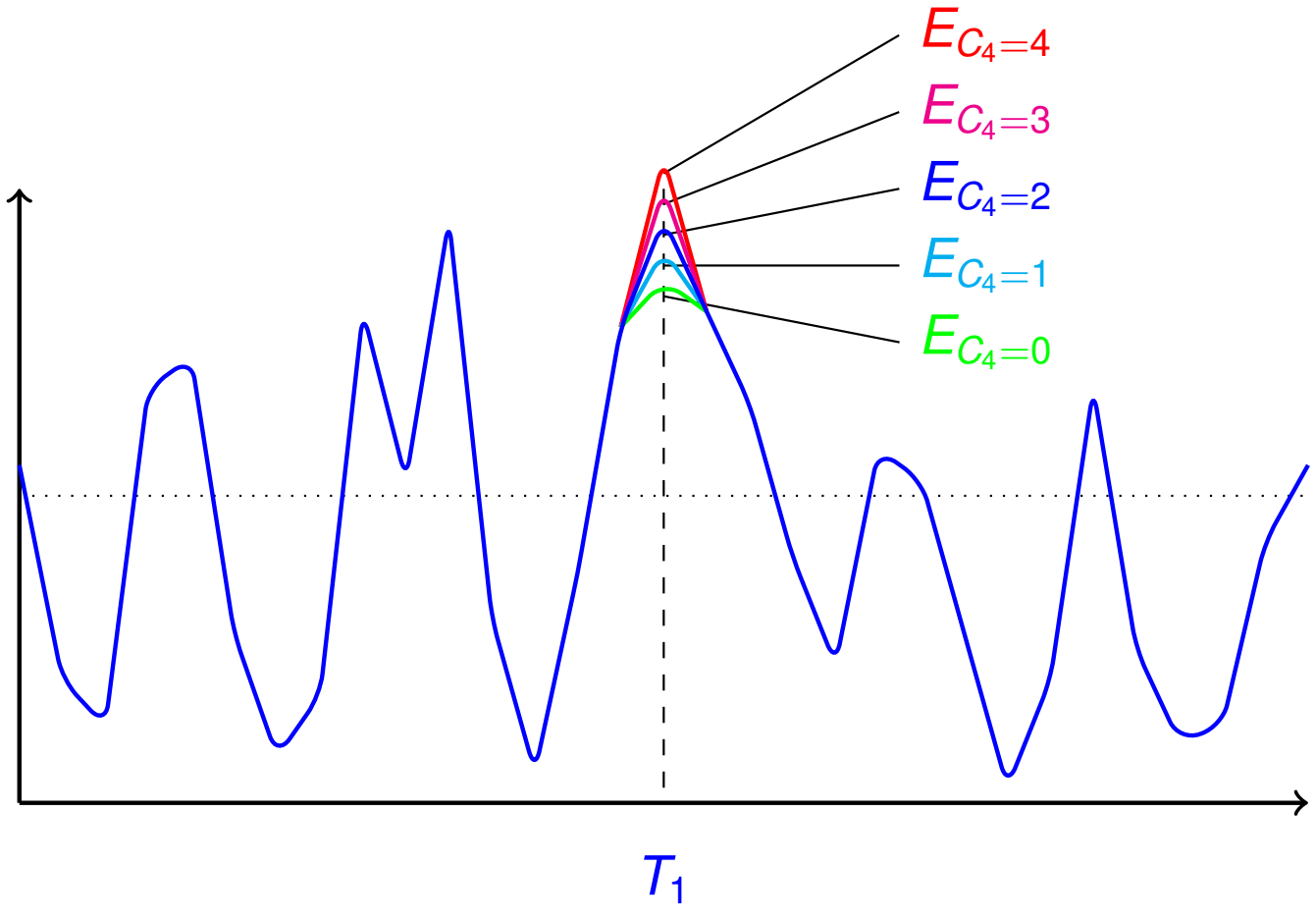
$$C_{moy_{C_4=j}}(T_1) = j + \bar{B}$$

- La formule de calcul de la trace différentielle fait ressortir cette cohérence pour l'échantillon correspondant à T_1 :

$$\begin{aligned} C_{diff}(T_1) &= -2 \times C_{moy_{C_4=0}}(T_1) - C_{moy_{C_4=1}}(T_1) + C_{moy_{C_4=3}}(T_1) + 2 \times C_{moy_{C_4=4}}(T_1) \\ &= -2 \times (0 + \bar{B}) - (1 + \bar{B}) + (3 + \bar{B}) + 2 \times (4 + \bar{B}) \\ &\approx 10 \end{aligned}$$

Exemple : DPA sur DES

Mode opératoire de l'attaque



Exemple : DPA sur DES

Pourquoi l'attaque fonctionne ? (hypothèse erronée)

- On suppose maintenant que l'on fait une hypothèse fausse sur S ($S_H \neq S_T$)
- Lorsque l'on applique le modèle de consommation, ce dernier ne prédit pas correctement le comportement des 4 bits étudiés du registre R_i
- De ce fait, le partitionnement effectué sur l'ensemble des traces est incohérent par rapport au comportement réel de ces 4 bits : pour $j = 0, \ldots, 4$, $\forall C_{mesurée}(P_i, K, t) \in E_{C_4=j}$, on a :

$$C_{mesurée}(P_i, K, T_1) = k_i + B$$

avec $k_i \in [\![0 \ldots 4]\!]$

Exemple : DPA sur DES

Pourquoi l'attaque fonctionne ? (hypothèse erronée)

- Du fait de l'incohérence du partitionnement, la trace moyenne dans chaque partition est identique :

$$C_{moy_{C_4=j}}(T_1) = 2 + \bar{B}$$

- La formule de calcul de la trace différentielle donne un résultat proche de 0 :

$$\begin{aligned} C_{diff}(T_1) &= -2 \times C_{moy_{C_4=0}}(T_1) - C_{moy_{C_4=1}}(T_1) + C_{moy_{C_4=3}}(T_1) + 2 \times C_{moy_{C_4=4}}(T_1) \\ &= -2 \times (2 + \bar{B}) - (2 + \bar{B}) + (2 + \bar{B}) + 2 \times (2 + \bar{B}) \\ &\approx 0 \end{aligned}$$

- Ce phénomène se produit aussi pour les échantillons ne correspondant pas à T_1, que l'hypothèse sur S soit correcte ou pas

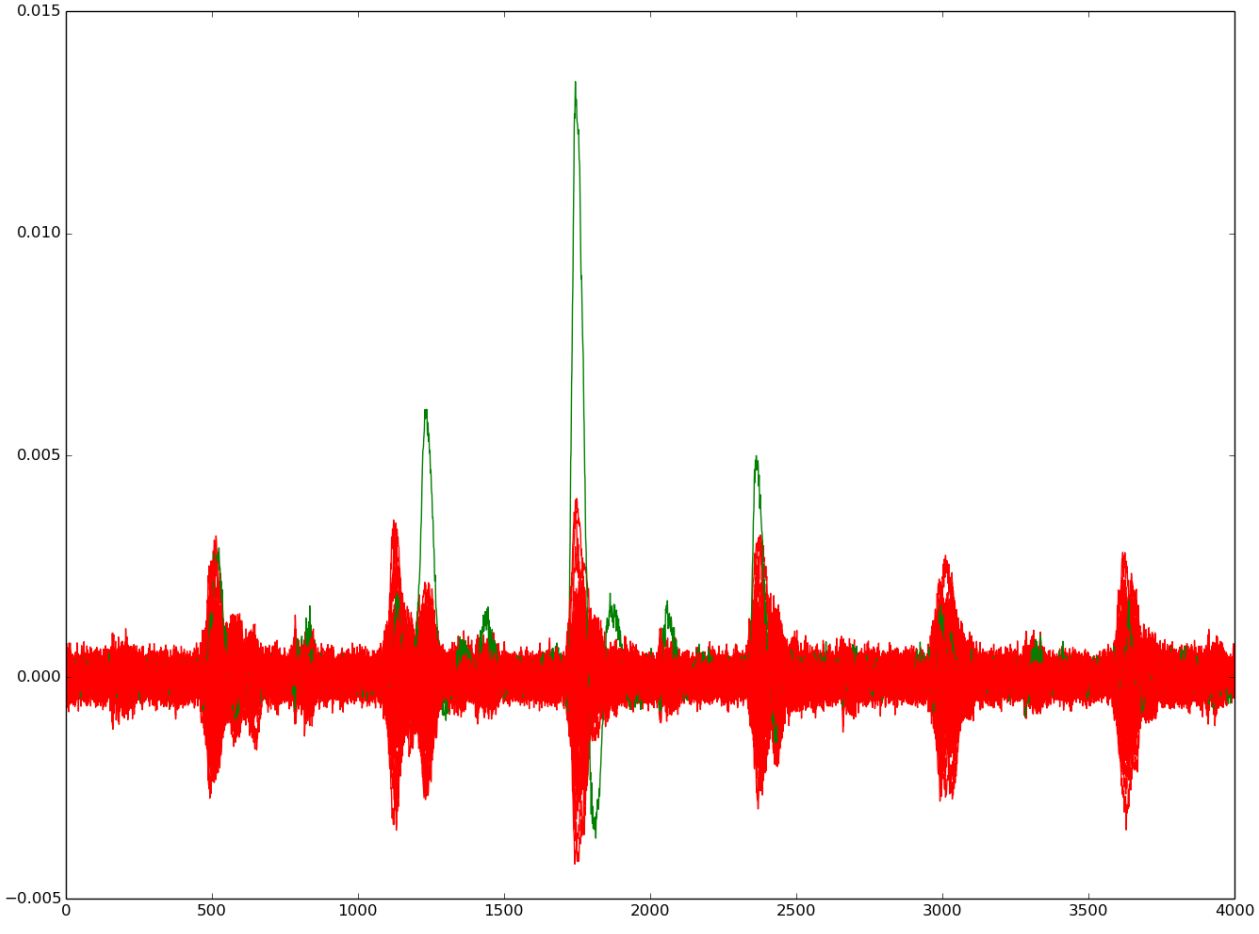
Exemple : DPA sur DES

Pourquoi l'attaque fonctionne ?

- En conclusion, tous les échantillons des traces différentielles sont nuls à l'exception de celui correspondant à l'instant T_1 pour l'hypothèse correcte de S

Exemple : DPA sur DES

Exemple de trace différentielle



- 64 traces différentielles superposées (63 en rouge et 1 en verte) pour la SBox 2

Exemple : DPA sur DES

Derniers points

- On a retrouvé 6 bits de SK_0 qui nous donnent directement 6 bits de K
- Si on répète l'attaque sur les autres SBox, on peut retrouver les 48 bits de SK_0 et donc 48 bits de K
- Les 8 bits restants peuvent être retrouvés en attaquant les tours suivants (maintenant que le premier tour est entièrement connu) ou plus simplement en faisant une recherche exhaustive
- Complexité totale de l'attaque : 64 hypothèses pour chacune des 8 Sbox plus la recherche exhaustive : $64 \times 8 + 256$ opérations

Outlines

Exemples de modèles de fuite

- Poids de Hamming : $M(S) = \mathrm{HW}(S)$
  - Adapté dans le cas de bus mis à zéro
- Distance de Hamming [3] :
  $M(S) = \mathrm{HD}(S, S_{-1}) = \mathrm{HW}(S \oplus S_{-1})$
  - Adapté pour une implémentation matérielle (consommation du CMOS)
- Switching distance [13] : $M(S) = 1$ si transition $0 \rightarrow 1$, $(1 - \delta)$ si transition $1 \rightarrow 0$, 0 sinon
  - Adapté pour une analyse EM en champ proche

Exemples de distingueurs

Classification par [14]

- Partitionnement
  - Différence de moyennes [9] : DPA
  - Covariance [1]
  - Information mutuelle [6] : MIA
- Comparaison
  - Corrélation [3] : CPA

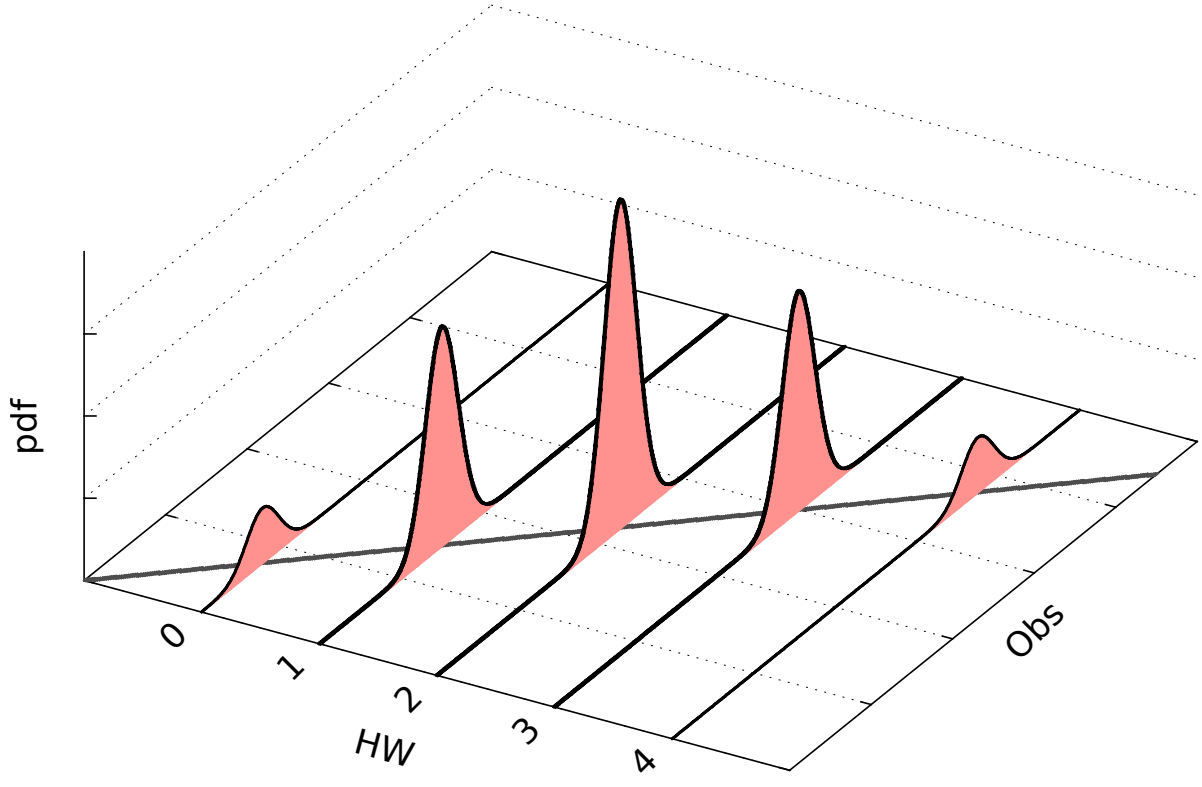
Correlation Power Analysis (CPA)

Coefficient de corrélation de PEARSON

$$\rho_{X,Y} = \frac{\mathrm{cov}(X, Y)}{\sigma_X \sigma_Y}$$

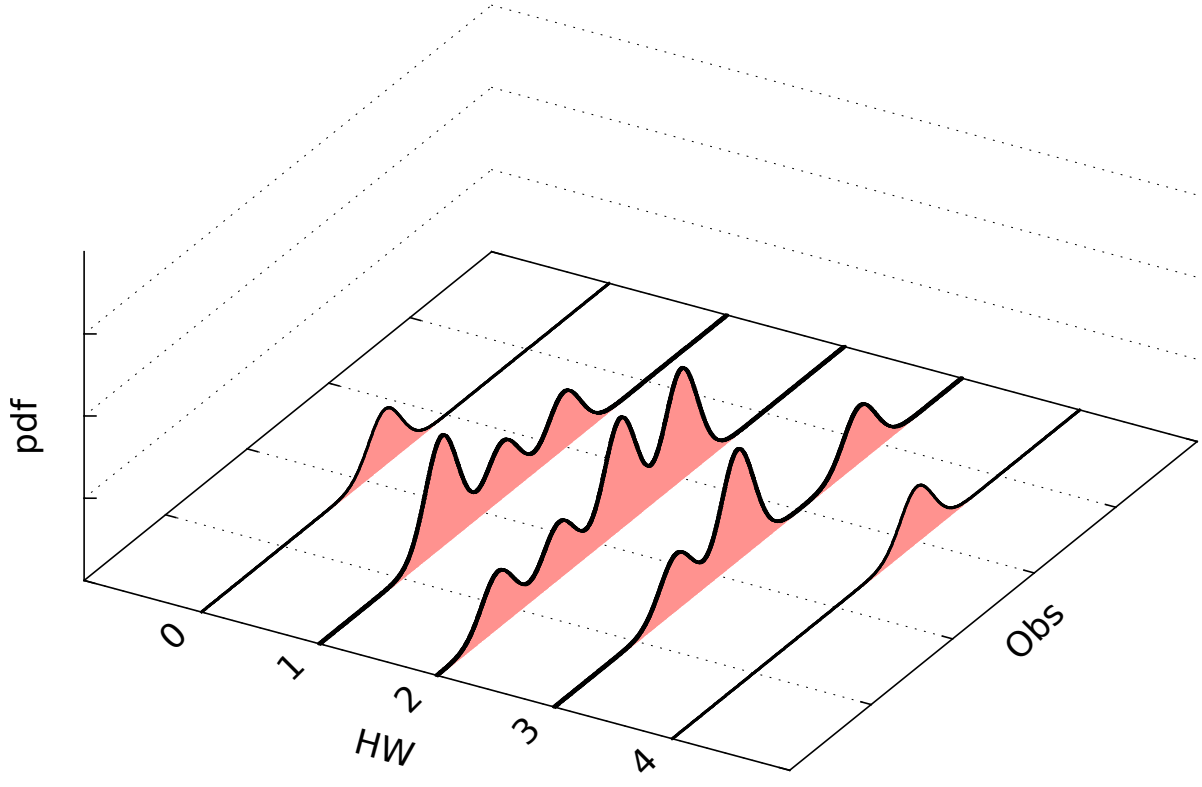
- Si le comportement du circuit est linéaire par rapport à la prédiction du modèle de consommation, il est intéressant d'utiliser le coefficient de corrélation linéaire pour tester la validité de l'hypothèse

Correlation Power Analysis (CPA)



Bonne hypothèse de clé, corrélation $\neq 0$

Correlation Power Analysis (CPA)



Mauvaise hypothèse de clé, corrélation $= 0$

Attaques template [5]

Principe

- Si l'on dispose d'un second circuit, identique à celui que l'on veut attaquer, mais que l'on contrôle totalement, on peut réaliser une attaque template
- Le principe est d'apprendre (profiler) comment fuit le circuit avant d'utiliser cette connaissance sur le circuit cible pour l'attaquer en très peu de traces
- Il y a donc deux étapes
  1. Une étape de profilage réalisée sur le circuit test
  2. Une étape d'attaque réalisée sur le circuit cible

Attaques template

Profilage

On suppose que le circuit effectue une opération parmi K : $O_1, \ldots, O_K$ (exemple : manipulation d'une variable sensible)

1. Collecter de nombreuses traces du circuit test pour chacune des K opérations $O_1, \ldots, O_K$
2. Calculer les traces moyennes : $M_1, \ldots, M_K$
3. Optionnel : calculer les différences entre les traces moyennes $M_1, \ldots, M_K$ pour identifier les points d'intérêt $P_1, \ldots, P_N$

Attaques template

Profilage

4. Pour chaque opération O_i
   - 4.1 Pour chaque trace T pour cette opération O_i, le vecteur de bruit pour cette trace est :

$$N_i(T) = (T[P_1] - M_i[P_1], \ldots, T[P_N] - M_i[P_N])$$

   - 4.2 Calculer la matrice de covariance de bruit :

$$\Sigma_{N_i}[u, v] = \mathrm{cov}(N_i[P_u], N_i[P_v])$$

   - 4.3 Le template pour O_i est (M_i, Σ_{N_i})

Attaques template

Attaque

Étant donné une observation S du circuit cible

1. Pour chaque opération hypothétique O_i
   1.1 Calculer le vecteur de bruit

$$\mathbf{n} = N_i(S) = (S[P_1] - M_i[P_1], \ldots, S[P_N] - M_i[P_N])$$

   1.2 Calculer la probabilité d'observer $\mathbf{n}$ (loi normale multidimensionnelle)

$$p_{N_i}(\mathbf{n}) = \frac{1}{\sqrt{(2\pi)^N |\Sigma_{N_i}|}} \exp(-\frac{1}{2}\mathbf{n}^T \Sigma_{N_i}^{-1} \mathbf{n})$$

   où $|\Sigma_{N_i}|$ est le déterminant de Σ_{N_i}
2. L'opération la plus probable est celle pour laquelle la probabilité d'observer $\mathbf{n}$ est la plus importante

Attaques template

Améliorations

- Il est possible d'utiliser l'analyse en composantes principales (PCA) afin de réduire la taille des *templates*
- Les attaques *templates* sont très puissantes et peuvent bien souvent retrouver l'intégralité du secret en utilisant une ou quelques traces

Outlines

Attaques en temps de calcul

- Les attaques utilisant la consommation ou le rayonnement EM nécessitent un accès physique au composant visé
- En revanche, les attaques se basant sur le temps de calcul (*Timing Attacks*) peuvent être réalisées à distance, y compris à travers un réseau
- Exemples :
  - Récupération de clés à distance via le réseau [4]
  - Récupération de clés à partir d'une autre machine virtuelle [7]
- Sources des variations temporelles :
  - Algorithme
  - Mécanismes d'optimisation du processeur : cache, pipeline...

Attaques en temps de calcul

Exemple : Attaque de RSA via le réseau [4]

- Attaque du RSA d'OpenSSL (version 0.9.7)
- Les optimisations utilisées dans cette bibliothèque (théorème du reste chinois, exponentiation avec fenêtre glissante, réduction de Montgomery, multiplication Karatsuba) rendent le temps de calcul légèrement dépendant de la clé secrète
- Les auteurs montrent que l'attaque peut être menée en local ou même à distance à travers le réseau
- En effet, moyennant de nombreux essais, la latence et la gigue introduite par le réseau (notamment les switchs) ne suffisent pas à masquer les faibles variations du temps de calcul
- Ici, la source de l'attaque provient de l'algorithme et de son implémentation

Attaques en temps de calcul

Exemple : Attaque d'AES cross-VM [7]

- La source des variations sur le temps de calcul peut également provenir des mécanismes d'optimisation internes au processeur (caches, pipeline, prédiction de branchement, etc.)
- Ces attaques peuvent par exemple être utilisées pour voler des informations sensibles appartenant à une autre machine virtuelle s'exécutant sur la même machine physique (très grave dans un contexte de cloud computing)
- Rappels :
  - *Cache* : mémoire rapide qui enregistre temporairement les données en provenance de la mémoire
  - Plus rapide que la mémoire mais en quantité plus limitée
  - Exploite les principes de localité spatiale et temporelle d'un programme

Attaques en temps de calcul

Exemple : Attaque d'AES cross-VM [7]

- Principe de base de l'acquisition (prime+probe)
  - L'adversaire rempli le cache avec ses données (en faisant simplement des accès mémoires aux bonnes adresses)
  - Le processus cible s'exécute : les lectures qu'il effectue remplacent certaines lignes de cache
  - L'adversaire reprend la main et vérifie quelles lignes du cache ont été évincées (en accédant de nouveau aux différentes adresses et en vérifiant le temps d'accès : court = donnée toujours dans le cache, long = donnée évincée)
  - $\rightsquigarrow$ L'adversaire obtient ainsi une partie des adresses auxquelles le processus cible accède
- Exploitation sur AES
  - Ces adresses permettent de savoir quelles parties des SBox (opération SubBytes) sont accédées et donc d'avoir de l'information sur l'état interne de l'algorithme

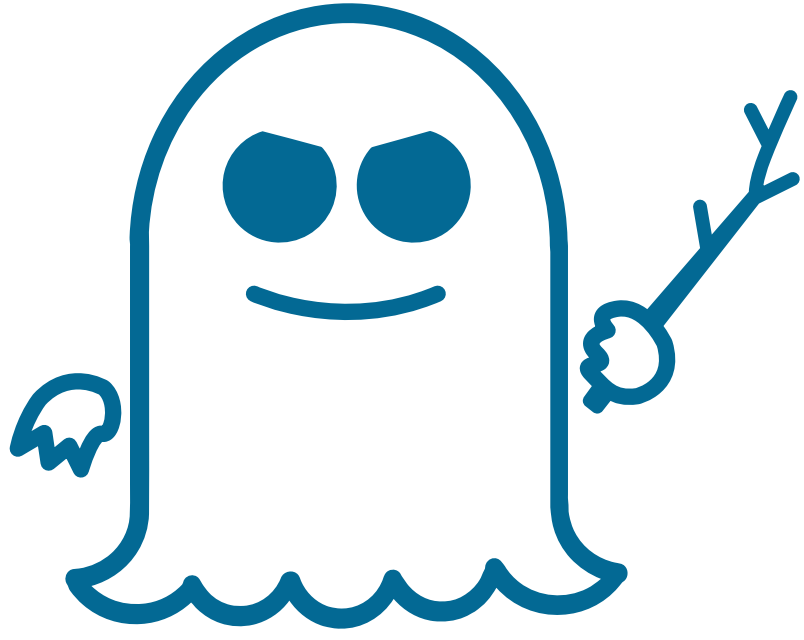
Attaques en temps de calcul

Exemple : Attaque d'AES cross-VM [7]

- Les détails sont un peu plus compliqués à cause
  - De l'intervention de la MMU
  - De la hiérarchie des caches (3 niveaux L1, L2, L3)
  - De l'intervention de l'OS (changement de contexte notamment)
  - ...
- Lire l'article pour plus de détails

Attaques Micro-architecturales

Meltdown [10] et Spectre [8]



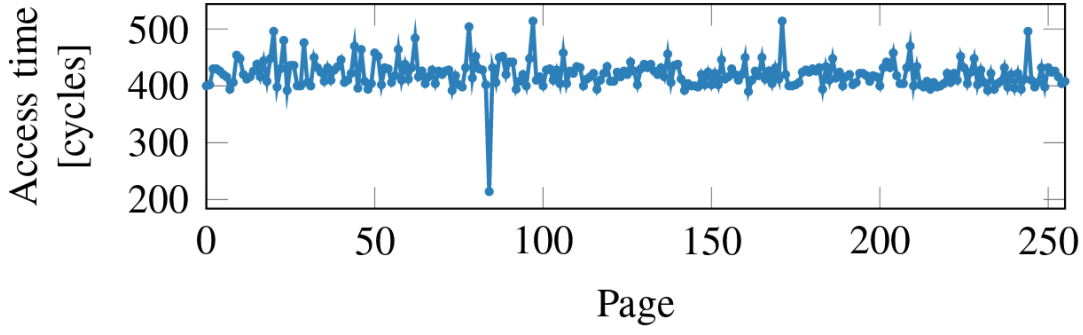
[https://meltdownattack.com/]

Meltdown : Principes de l'attaque

- Execution speculative utilisé dans tout processeur moderne (Intel, AMD, ARM, . . .)
- Pas (ou moins) de contrôle d'accès mémoire pendant la speculation
- Résidus d'accès spéculatifs dans le cache
- Utilisation d'un canal auxiliaire (cache) pour les rendre visible
- Attaque très puissante qui ne laisse pas de traces !

Meltdown : Attack program

```
1  raise_exception();
2
3  // This code should never be executed:
4  access(probe_array[ data * 4096 ]);
```



- Accès spéculatif au secret `data`
- Accès à `probe_array` à une position dépendant du secret
- Reconstruction de la valeur de `data` par le temps d'accès au différentes pages de mémoire de `probe_array`

Outlines

Rendre l'attaque difficile

Pseudo contre-mesures

- Ces contre-mesures rendent l'attaque plus difficile (mais pas impossible)
- Exemples
  - Générateur de bruit
  - Désynchronisation des traces
    - Horloge variable
    - Insertion d'instructions inutiles
- Il existe des techniques pour resynchroniser les traces et pour les débruiter

Contre-mesures

Différents niveaux

- Algorithme / Protocole
- RTL (*Register-Transfer Level*)
- Netlist / implémentation

Protocole

Changement de clés

- S'il faut 200 opérations de chiffrement pour que l'attaquant retrouve la clé, il suffit de changer la clé tous les 100 chiffrements
- Nécessite une connaissance du nombre de traces nécessaires pour effectuer l'attaque
- À l'extrême, on peut changer la clé à chaque opération de chiffrement

Masquage

Masking

- Masquer la variable sensible à l'aide d'une variable aléatoire
- Fonction initiale : $R = f(S)$ où S est une variable sensible
- Fonction masquée : $S' = S \oplus M$, $R' = g(S')$, $M' = h(M)$ tel quel $R = R' \oplus M'$ (exemple de masquage booléen)
- M est choisi aléatoirement à chaque opération de chiffrement (masque)
- On doit faire attention à ce qu'à aucun moment dans le calcul on ne manipule S directement (démasquage)
- Si l'adversaire retrouve S', il n'apprend aucune information sur S
- Pour attaquer, l'adversaire doit retrouver S' et M' en même temps (attaque d'ordre 2)

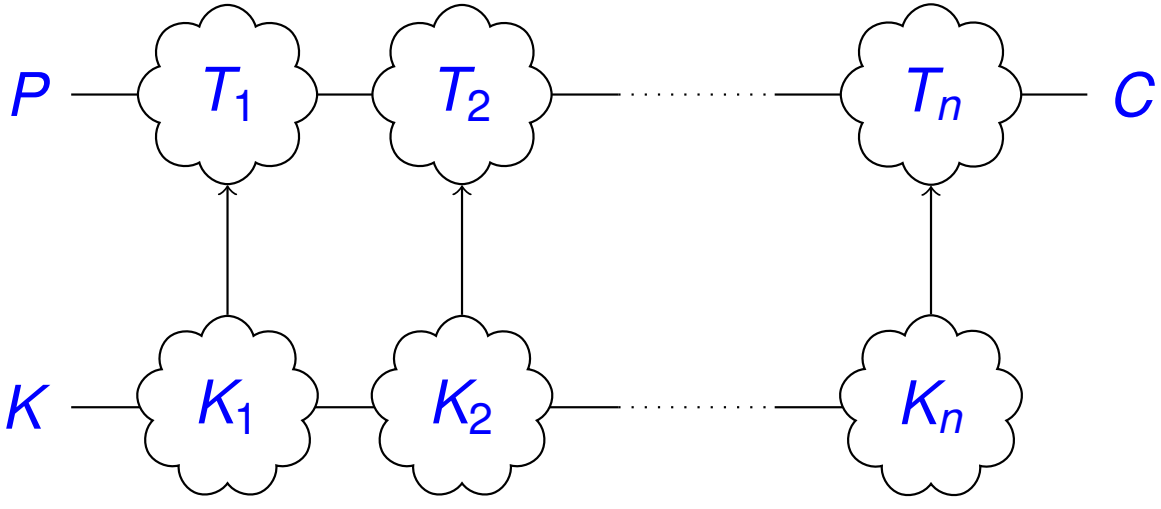
Masquage

Masking

- Pour augmenter la résistance, il est possible d'utiliser plusieurs masques
- Exemple : *Threshold Implementation* (TI) [12]

Implémentation déroulée [2]

- Il est possible de rendre les attaques en mesure de consommation plus difficile en utilisant une implémentation partiellement ou entièrement déroulée au lieu d'une implémentation itérative



Équilibrage

Hiding

- Faire en sorte que le comportement du système sur le canal auxiliaire considéré soit constant
  - Temps de calcul constant
  - Consommation identique
- Très difficile concernant le rayonnement EM (localité)

Équilibrage

Exemple sur RSA

Inputs : M , K
$R = 1$; $S = M$;
for $i = |K| - 1$; $i \geq 0$; $i - -$ **do**
    `/* Balanced branching */`
    if $K_i == 1$ **then**
        $R = R \times S$; $S = S^2$;
    else
        $S = S \times R$; $R = R^2$;
    end if
end for
Return $R = M^K$;

- Calcul de l'exponentiation modulaire par l'algorithme Montgomery ladder exponentiation

Équilibrage

Logique Dual-Rail avec précharge (DPL) [15, 11]

- Chaque variable booléenne a est représentée par deux signaux a_T et a_T

a_T	a_F	État	a
0	0	NULL0	-
0	1	VALID0	0
1	0	VALID1	1
1	1	NULL1	-

- Une fonction logique DPL $(s_T, s_F) = f((a_T, a_F), (b_T, b_F))$ doit respecter les propriétés suivantes :
  - Si a et b sont dans un état NULL0, s est dans un état NULL0
  - Si a et b sont dans un état VALID, s est dans un état VALID

Équilibrage

Logique Dual-Rail avec précharge (DPL)

- Exemple d'une fonction ET logique
  - $s_T = a_T \cdot b_T$
  - $s_F = a_F + b_F$
- Précharge : le calcul alterne les phases NULL0 et les phases valides
- On a ainsi pour chaque signal les transitions suivantes
  - $(0,0) \rightarrow (0,1) \rightarrow (0,0)$
  - $(0,0) \rightarrow (1,0) \rightarrow (0,0)$
- Donc à chaque phase, on a un signal et un seul qui change d'état $\rightsquigarrow$ consommation identique

Équilibrage

Problèmes

- Early evaluation : Problème si $f(VALID, NULL) = VALID$ (fuite d'informations si les deux signaux n'arrivent pas en même temps)
- Les deux réseaux (*true* et *false*) doivent être situés à proximité pour éviter trop de dispersion au niveau des temps de propagation et de la consommation des portes

Outlines

Conclusion

- Les attaques SCA exploitent les fuites d'informations liées à un comportement du système sur le canal auxiliaire considéré dépendant des opérations effectuées ou des données manipulées
- Ces attaques nécessitent souvent un accès physique au système cible (mais pas toujours, notamment les attaques *timing*)
- Ces attaques peuvent être particulièrement efficaces, notamment contre des circuits non protégés

Références I

[1] Régis Bevan and Erik Knudsen.
Ways to Enhance Differential Power Analysis.
In *ICISC*, volume 2587 of *Lecture Notes in Computer Science*, pages 327–342. Springer, 2002.

[2] Shivam Bhasin, Sylvain Guilley, Laurent Sauvage, and Jean-Luc Danger.
Unrolling Cryptographic Circuits : A Simple Countermeasure Against Side-Channel Attacks.
In *RSA Cryptographers' Track, CT-RSA*, volume 5985 of *LNCS*, pages 195–207. Springer, March 1-5 2010.
San Francisco, CA, USA. DOI : 10.1007/978-3-642-11925-5_14.

[3] Éric Brier, Christophe Clavier, and Francis Olivier.
Correlation Power Analysis with a Leakage Model.
In *CHES*, volume 3156 of *LNCS*, pages 16–29. Springer, August 11–13 2004.
Cambridge, MA, USA.

[4] David Brumley and Dan Boneh.
Remote timing attacks are practical.
In *Proceedings of the 12th Conference on USENIX Security Symposium*, pages 1–14, 2003.

[5] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi.
Template Attacks.
In *CHES*, volume 2523 of *LNCS*, pages 13–28. Springer, August 2002.
San Francisco Bay (Redwood City), USA.

[6] Benedikt Gierlichs, Lejla Batina, Pim Tuyls, and Bart Preneel.
Mutual information analysis.
In *CHES, 10th International Workshop*, volume 5154 of *Lecture Notes in Computer Science*, pages 426–442.
Springer, August 10-13 2008.
Washington, D.C., USA.

Références II

[7] Gorka Irazoqui, Thomas Eisenbarth, and Berk Sunar.
S$a : A shared cache attack that works across cores and defies vm sandboxing — and its application to aes.
In *2015 IEEE Symposium on Security and Privacy (SP)*, pages 591–604, May 2015.

[8] Paul Kocher, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, and Yuval Yarom.
Spectre attacks : Exploiting speculative execution.
ArXiv e-prints, January 2018.

[9] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun.
Differential Power Analysis.
In *Proceedings of CRYPTO'99*, volume 1666 of *LNCS*, pages 388–397. Springer-Verlag, 1999.
(PDF).

[10] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, and Mike Hamburg.
Meltdown.
ArXiv e-prints, January 2018.

[11] Maxime Nassar, Shivam Bhasin, Jean-Luc Danger, Guillaume Duc, and Sylvain Guilley.
BCDL : A high performance balanced DPL with global precharge and without early-evaluation.
In *DATE'10*, pages 849–854. IEEE Computer Society, March 8-12 2010.
Dresden, Germany.

[12] Svetla Nikova, Christian Rechberger, and Vincent Rijmen.
Threshold Implementations Against Side-Channel Attacks and Glitches.
In *Information and Communications Security : 8th International Conference, ICICS 2006, Raleigh, NC, USA, December 4-7, 2006. Proceedings*, pages 529–545. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

Références III

[13] Éric Peeters, François-Xavier Standaert, and Jean-Jacques Quisquater.
Power and electromagnetic analysis : Improved model, consequences and comparisons.
Integration, The VLSI Journal, special issue on "Embedded Cryptographic Hardware", 40 :52–60, January 2007.
DOI : 10.1016/j.vlsi.2005.12.013.

[14] François-Xavier Standaert, Benedikt Gierlichs, and Ingrid Verbauwhede.
Partition vs. Comparison Side-Channel Distinguishers : An Empirical Evaluation of Statistical Tests for Univariate Side-Channel Attacks against Two Unprotected CMOS Devices.
In *ICISC*, volume 5461 of *LNCS*, pages 253–267. Springer, December 3-5 2008.
Seoul, Korea.

[15] Kris Tiri and Ingrid Verbauwhede.
A Logic Level Design Methodology for a Secure DPA Resistant ASIC or FPGA Implementation.
In *DATE'04*, pages 246–251. IEEE Computer Society, February 2004.
Paris, France. DOI : 10.1109/DATE.2004.1268856.