Side-Channel Attacks : Sécurité et Attaques par Canaux Auxiliaires

Introduction, Systèmes cryptographiques et consommation, Chiffrement symétrique

Quentin Meunier

2024

Sorbonne Université
Laboratoire d'Informatique de Paris 6
4 Place Jussieu, 75252 Paris, France

Plan

Introduction à l'UE

Systèmes cryptographiques et consommation électrique

Rappels sur le chiffrement symétrique et l'AES

Algorithmes cryptographiques et implémentation

Les protocoles cryptographiques reposent sur des algorithmes mathématiques supposés sûrs

- Difficile/Impossible de retrouver une information secrète dans un temps raisonnable

Les implémentations des algorithmes prouvés sûrs peuvent être vulnérables

- Attaques liées au logiciel et au matériel sur lesquels les algorithmes sont déployés

Objectif de l'UE : montrer comment des informations obtenues ou des fautes injectées lors de l'exécution d'une implémentation peuvent permettre de retrouver les secrets de protocoles cryptographiques et mettre à mal la sécurité escomptée

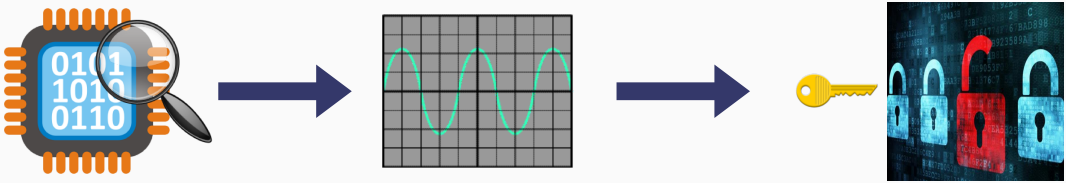
- Éléments sur la sécurité, principe des attaques par canal auxiliaire et par injection de fautes
- Exemples d'attaques par canal auxiliaire : AES, RSA, VerifyPIN
- Exemple de contre-mesures logicielles et matérielles
- Réalisation et simulation d'attaques physiques

Classification des attaques physiques

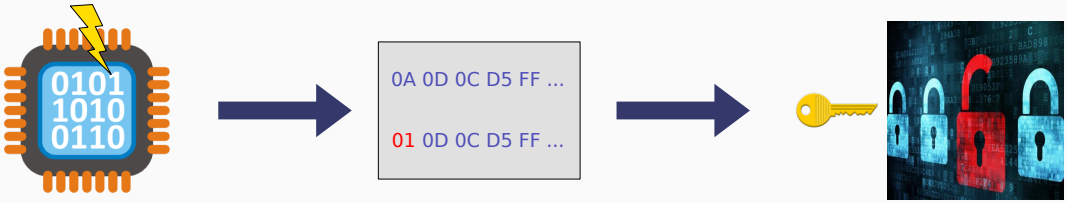
- Attaques invasives
  - Depackaging de la puce
  - Station de mesure pour observer des signaux en particulier
  - Très puissantes mais très couteuses
- Attaques semi-invasives
  - Depackaging de la puce mais pas de contact électrique avec la surface
  - Utilisation de rayons X, d'un champ électromagnétique ou de lumière (laser) pour induire une faute
  - $\Rightarrow$ En déduire des informations secrètes ou contourner une protection
  - Cout du matériel et de l'expertise élevé
- Attaques non-invasives
  - Attaque du système tel quel, pas de traces de l'attaque après coup
  - Cout relativement faible en comparaison

Attaques non-invasives : 2 catégories

- Attaques passives / par observation : temps, puissance dissipée, émission électromagnétique (EM)



- Attaques actives / par perturbation : glitch d'horloge ou de tension, impulsion EM



- Dans cette UE, concentration sur les attaques par observation du courant consommé

Évaluation de l'UE

Rapports de TME et travail personnel

- TME utilisant la mesure de consommation pour attaquer :
  - L'authentification par code PIN
  - L'AES par DPA, CPA et Attaque par template
- Remise de 3 rapports sur le travail réalisé en TME
- Ces notes comptent pour 50% de la note finale

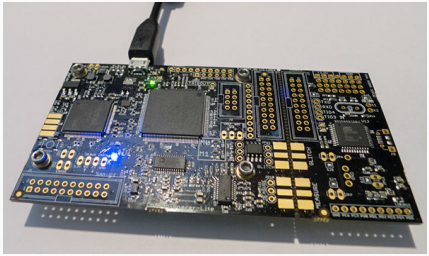
Examen final : début février

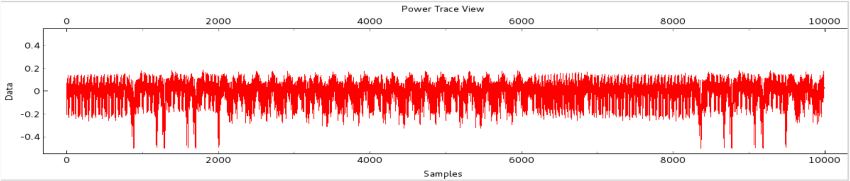
- Réalisé à partir de questions en lien avec les cours + TME
- 50% de la note finale

Déroulement de l'UE : Cours et TME

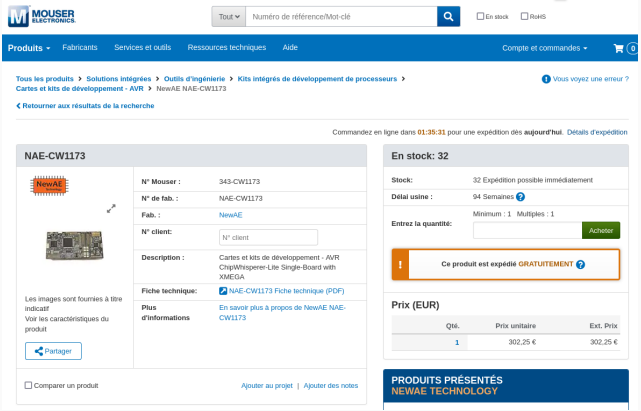
- *20/09* Cours Quentin Meunier : Introduction à l'UE et aux attaques par canaux auxiliaires
- *27/09* TME1 : Prise en main de la carte ChipWhisperer
- *04/10* Cours Adrian Thillard : Attaques par canaux auxiliaires
- *11/10* TME2 : Attaque de la vérification d'un mot de passe
- *18/10* Cours Quentin Meunier : Attaques différentielles et template + TME3 : Influence du HW
- *25/10* TME4 : DPA Mono-bit et Multi-bit
- *22/11* TME5 : Attaque CPA
- *29/11* Cours Quentin Meunier : Masquage et analyse du masquage + TME6
- *06/12* TME7 : Masquage, schéma de Herbst
- *13/12* TME8 : Attaque par template
- *10/01* Cours Karine Heydemann + David Vigilant : Fautes, protections et compilation (probablement déplacé dans la semaine)
- *17/01* TME9 : Attaque en faute différentielle (exploitation)
- *24/01* Cours Jessy Clédière : Injection de faute : attaques et contre-mesures

Carte pour les TME : Chipwhisperer





Carte pour les TME : Chipwhisperer



- Carte achetée à 302.25 euros l'unité
- Une carte par personne prêtée au début de l'UE
- Vous êtes responsables de la rendre en bon état à la fin de l'UE, sous peine de remboursement

Informations pratiques concernant les TME

- Les TME peuvent être faits sur votre machine personnelle ou sur les machines de l'université, et nécessitent l'utilisation d'une machine virtuelle sur VirtualBox
- Peut arriver que cela ne marche pas sur votre machine personnelle (adaptateur mac, ...), dans ce cas utiliser les machines de l'université
- Si vous comptez faire le TME sur votre ordinateur, téléchargez la VM avant la première séance de TME ! (3 Go)
- Consignes des TME sur moodle

Dernières informations

- Transparents présentés par les intervenants mis sur moodle après les séances
- Regarder le planning CCA + MSI : des incohérences peuvent arriver, les 2 plannings devraient être identiques
- Les TME durent 2 ou 4h, ne pas se plaindre de la charge de travail en fin d'année ou de semestre, on est à votre disposition pendant ces créneaux
- Plagiat, partage de son code, récupération de code ou bout de rapport des années précédentes : 0 au TME correspondant
- Les rapports sont à soigner : être concis, expliquer ce qui a été fait, comment et pourquoi ; extraits de code dans le rapport et code complet à fournir à côté
- Référence pour le cours : *Power analysis attacks: Revealing the secrets of smart cards*, S. Mangard, E. Oswald, T. Popp (2007). *Springer Science & Business Media*.

Plan

Introduction à l'UE

Systèmes cryptographiques et consommation électrique

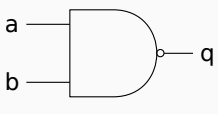
Rappels sur le chiffrement symétrique et l'AES

Composants des systèmes cryptographiques

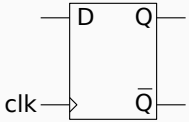
- Matériel dédié (ex : coprocesseur AES)
- Matériel généraliste (ex : processeur, micro-controlleur)
- Logiciel cryptographique (ex : implémentation logicielle de l'AES)
- Mémoire
- Interface

Cellules logiques

- Les cellules logiques sont les briques de base d'un circuit
- Cellules combinatoires, portes logiques (ex : NAND, XOR)
- Cellules séquentielles, registres (ex : Bascule D)



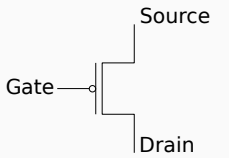
Cellule NAND
à 2 entrées



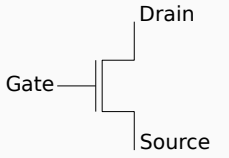
Bascule D active
sur front montant

Cellules logiques

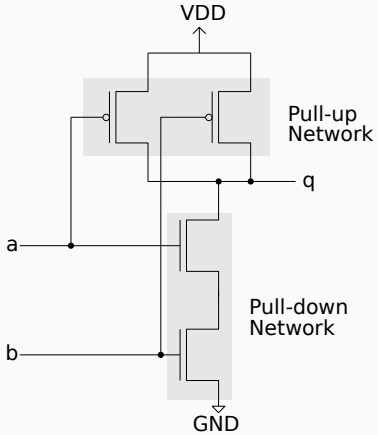
- Les cellules sont en général réalisées à l'aide de transistors de technologie CMOS
- Un transistor agit comme un interrupteur contrôlé par une tension
- La fonction connecte la sortie et le "1 Logique" (VDD) avec de PMOS
- Le complément de la fonction connecte la sortie et le "0 Logique" (GND) avec des NMOS



Transistor PMOS

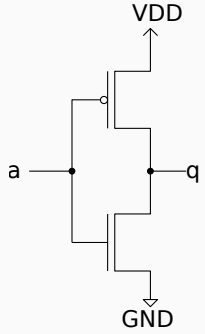


Transistor NMOS



Consommation électrique des cellules logiques

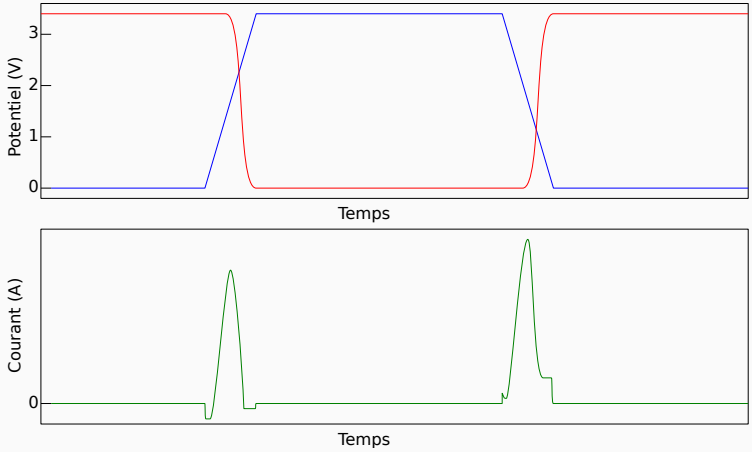
- 2 types de consommation
- Statique : lorsqu'il n'y a pas de changement d'état dans la cellule
- Dynamique : liée au changement d'état de la cellule ($\Leftrightarrow$ changement de la valeur en sortie)
- Causes de la consommation dynamique :
  - Chargement de la capacité de la cellule : capacité interne et externe (fils contrôlés par la sortie et entrées des transistors suivants)
  - Court-circuit pendant un court instant après un changement d'état : période pendant laquelle les deux transistors sont passants



Transition	Power consumption	Type of power consumption
$0 \rightarrow 0$	P_{00}	statique
$0 \rightarrow 1$	P_{01}	statique + dynamique
$1 \rightarrow 0$	P_{01}	statique + dynamique
$1 \rightarrow 1$	P_{11}	statique

- En général, on a $P_{00} \approx P_{11} \ll P_{01}, P_{10}$
- $\Rightarrow$ La consommation dynamique est prépondérante

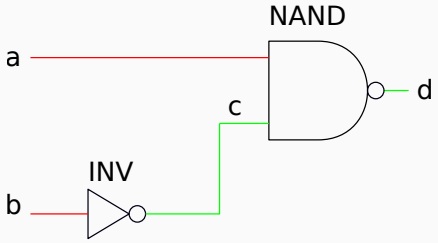
Consommation électrique des cellules logiques

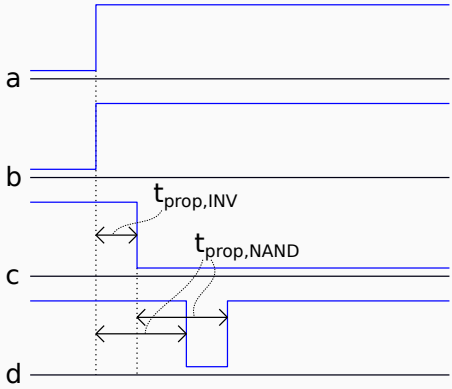


- Consommation (courant) pour un inverseur CMOS
- Pic de consommation un peu plus grand lors du changement d'état de la sortie de $0 \rightarrow 1$

Glitches

- La sortie d'une porte met un certain temps à changer d'état quand ses entrées changent
- $\Rightarrow$ Certaines portes peuvent avoir une valeur temporaire en sortie

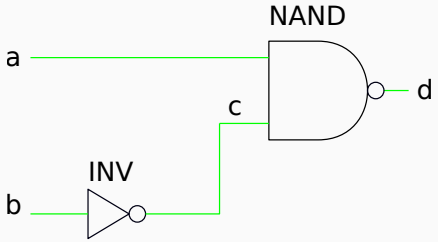


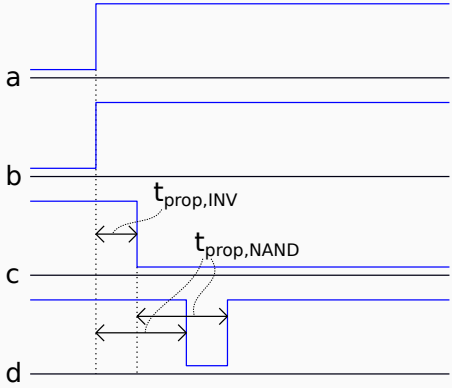


- Les glitches peuvent avoir un effet important sur la consommation

Glitches

- La sortie d'une porte met un certain temps à changer d'état quand ses entrées changent
- $\Rightarrow$ Certaines portes peuvent avoir une valeur temporaire en sortie

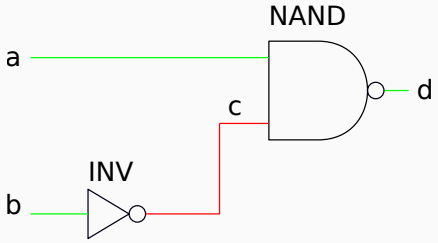


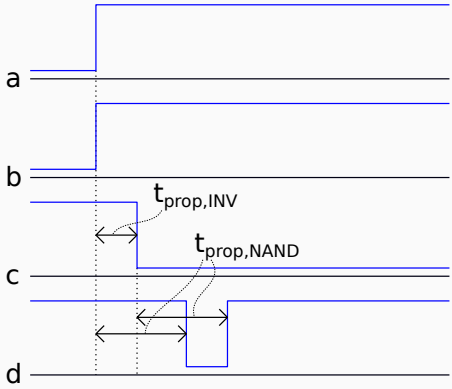


- Les glitches peuvent avoir un effet important sur la consommation

Glitches

- La sortie d'une porte met un certain temps à changer d'état quand ses entrées changent
- $\Rightarrow$ Certaines portes peuvent avoir une valeur temporaire en sortie

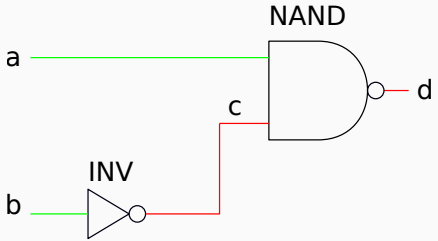


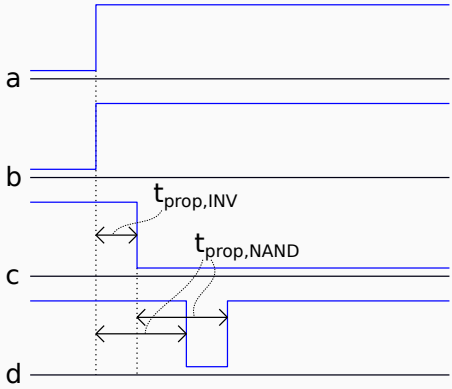


- Les glitches peuvent avoir un effet important sur la consommation

Glitches

- La sortie d'une porte met un certain temps à changer d'état quand ses entrées changent
- $\Rightarrow$ Certaines portes peuvent avoir une valeur temporaire en sortie

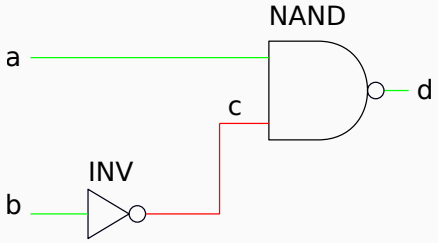


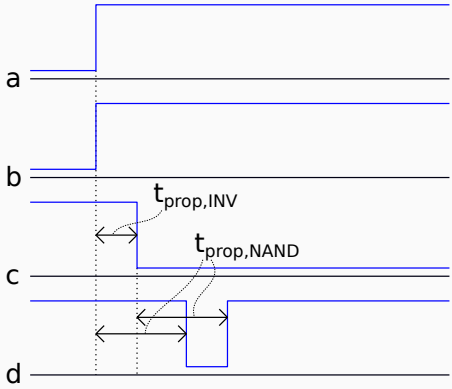


- Les glitches peuvent avoir un effet important sur la consommation

Glitches

- La sortie d'une porte met un certain temps à changer d'état quand ses entrées changent
- $\Rightarrow$ Certaines portes peuvent avoir une valeur temporaire en sortie





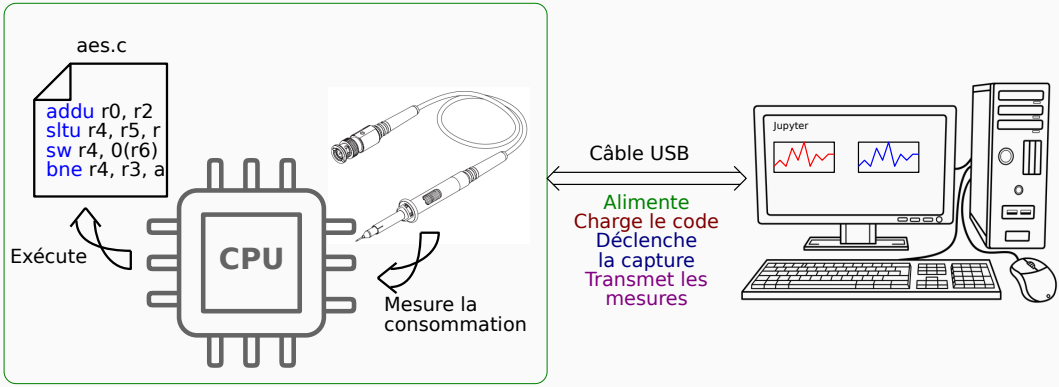
- Les glitches peuvent avoir un effet important sur la consommation

Modèles de consommation

- Modèles précis
  - Simulation analogique avec la netlist des transistors et les éléments (capacités) parasites du circuit
  - Simulation logique avec la netlist des cellules et des informations sur les temps de propagation
- Modèle simple : Distance de Hamming (HD)
  - Nombre de transitions $0 \rightarrow 1$ et $1 \rightarrow 0$ qui se produisent durant un certain laps de temps
  - Toutes les transitions de toutes les cellules ont le même poids dans le modèle
  - HD(v0, v1) = HW(v0 $\oplus$ v1) où HW est le poids de Hamming (nombre de bits à 1 dans le mot)
  - Exemple : HD(0xAE, 0x33) = 5
  - $\Rightarrow$ Modèle puissant (simple et proche de la réalité) qui peut-être utilisé par des attaquants qui ont connaissance de certaines parties du système (ex : bus, mémoire, registres)
- Si on ne dispose pas d'informations suffisantes sur l'architecture pour utiliser le modèle distance de Hamming, modèle Poids de Hamming (HW)
  - Équivalent à la distance de Hamming si tous les bits de v0 ou v1 sont égaux à 0 (ou 1)
  - Si tous les bits de v0 ou v1 sont constants mais différents, relation d'autant plus forte avec HD que le nombre de bits à 0 ou 1 est élevé
  - Si les bits de v0 ou v1 sont distribués uniformément : pas de relation avec HD
- Variations au modèle HD : affecter des poids différents aux différentes cellules, et aux transitions $1 \rightarrow 0$ et $0 \rightarrow 1$

Setup utilisé en TP

Carte ChipWhisperer Lite



Caractéristiques des oscilloscopes

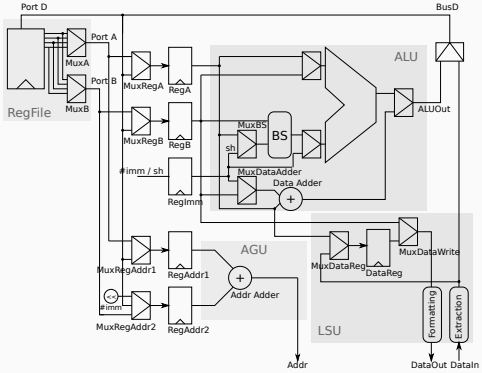
- Taux d'échantillonage : nombre de mesures de consommation qu'il est possible de faire par seconde
- Résolution : Nombre de valeurs possibles du signal après conversion d'une grandeur physique continue en grandeur numérique

Pour la carte ChipWhisperer Lite :

- Fréquence d'échantillonage de 29.48 MHz et résolution sur 10 bits
- Fréquence du processeur de 7.37 MHz
- $\Rightarrow$ 4 samples / cycle

Simple Power Attacks

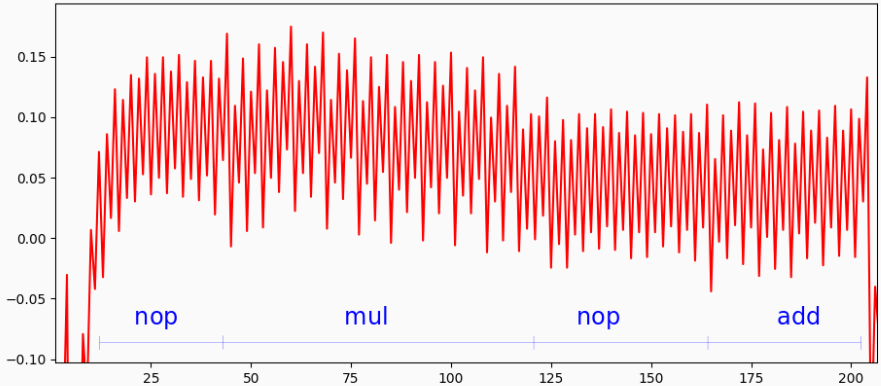
- Un processeur est fait de cellules logiques combinatoires et de registres
- Lors de l'exécution d'une instruction, changement d'état de certaines cellules



- But d'une attaque : exploiter les changements d'état liés à une variable/information secrète
- Dans le cadre de l'UE : processeur = boite noire, on ne sait pas d'où vient la fuite, modèle de consommation = HW (si besoin d'un modèle)

Simple Power Attacks : profil de consommation des instructions

- Différentes instructions consomment différemment



Simple Power Attacks : chemin d'exécution

- Un chemin d'exécution, ou une trace d'exécution est la séquence de toutes les instructions exécutées par le processeur
- Varie d'une exécution à l'autre si des branchements conditionnels dépendent des entrées

```
1  uint8_t gf_times2(uint8_t a) {
2    uint8_t r;
3    if ((a >> 7) == 0) {
4      r = a << 1;
5    }
6    else {
7      r = (a << 1) ^ 0x1B;
8    }
9    return r;
10 }
```

- Code avec plusieurs chemins d'exécution selon la valeur de a

```
1  uint8_t gf_times2(uint8_t a) {
2    uint8_t r;
3    r = ((int8_t) a) >> 7;
4    r = r & 0x1B;
5    r = r ^ (a << 1);
6    return r;
7  }
```

- Code avec un unique chemin d'exécution

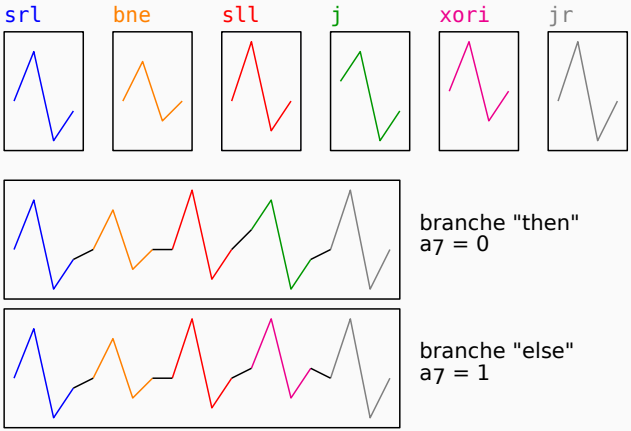
Simple Power Attacks : conditions

- Possible de retrouver la valeur d'une variable de condition if/else à partir de la trace d'instructions
- Exemple : bit de poids fort de a de la fonction `gf_times2`

- Code assembleur

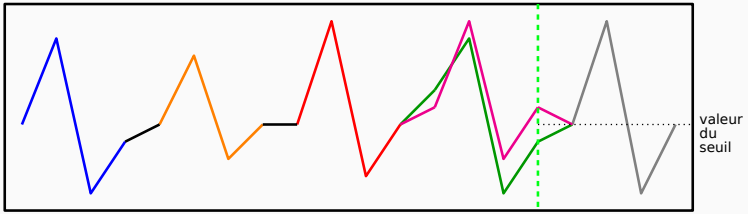
```
  srl r8, r8, 7
  bne r8, r0, _else
  sll r2, r8, 1
  j _endif
_else:
  sll r2, r8, 1
  xori r2, r2, 0x1B
_endif:
  jr r31
```

- Profil de consommation de ces instructions :



Simple Power Attacks : notion de seuil

- Comment discriminer les 2 cas ?
- En superposant les traces, on a :



- Si on connait le profil de consommation, on peut donc retrouver la valeur de la condition à partir d'un sample bien choisi et d'un seuil
- Il est aussi possible de recombiner les valeurs de plusieurs samples si cela ne suffit pas

Simple Power Attacks : équilibrage des branches

- Soit le code suivant

```
1   bool ok = true;
2   bool dummy = true;
3   for (int i = 0; i < size
        ; i+= 1) {
4     if (tab[i] != t[i]) {
5       ok = false;
6     }
7     else {
8       dummy = false;
9     }
10  }
```

- Code assembleur

```
_for:
  beq r4, r10,
_fin_for
  lw r8, 0(r4)
  lw r9, 0(r5)
  beq r8, r9, _else
  addiu r2, r0, 0
  j _endif
_else:
  addiu r15, r0, 0
_endif:
  addiu r4, r4, 4
  addiu r5, r5, 4
  j _for
```

- Trace d'exécution dans le cas "then" : beq, lw, lw, beq, addiu, j, addiu, addiu, j
- Trace d'exécution dans le cas "else" : beq, lw, lw, beq, addiu, addiu, addiu, j

- $\Rightarrow$ Équilibrer les branches if/else ne marche pas en général

Simple Power Attacks : exemple de l'exponentiation modulaire

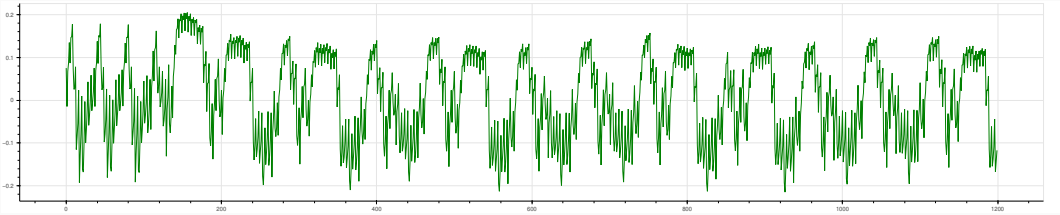
- Pseudo-code de l'exponentiation modulaire

```
1  mpi_t exp(mpi_t b, mpi_t e, mpi_t m) {
2    mpi_t res = 1;
3    for (int32_t i = 64; i >= 0; i -= 1) {
4      res = square(res);
5      res = modulo(res, m);
6      if (get_bit(e, i) == 1) {
7        res = mult(res, b);
8        res = modulo(res, m);
9      }
10   }
11   return res;
12 }
```

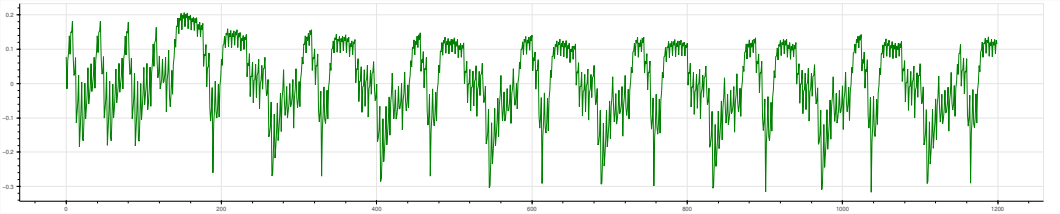
- $\Rightarrow$ La trace de consommation permet de révéler la valeur de tous les bits de l'exposant
- Dans RSA, ce calcul est effectué avec en exposant la clé secrète
- Les implémentations de ce calcul sont maintenant toujours sécurisées (multiply always)

Simple Power Attacks : exemple de l'exponentiation modulaire

- Square and Multiply

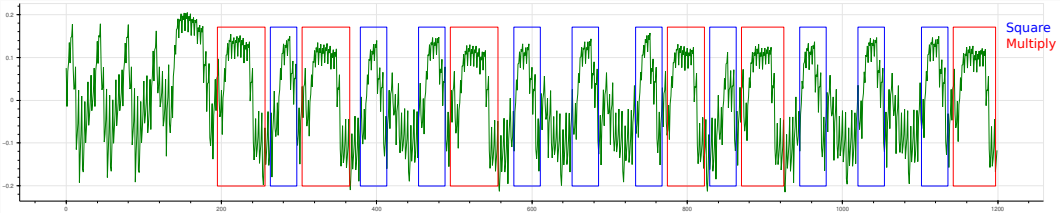


- Square and Multiply always

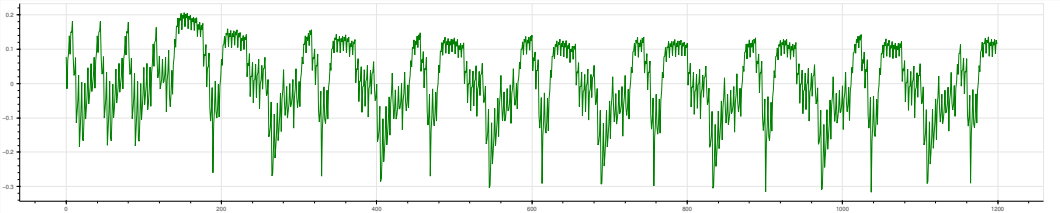


Simple Power Attacks : exemple de l'exponentiation modulaire

- Square and Multiply

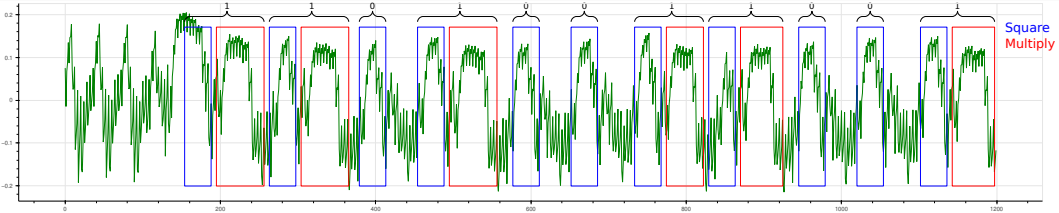


- Square and Multiply always

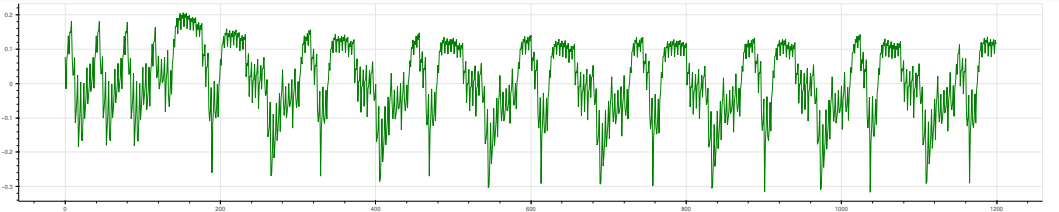


Simple Power Attacks : exemple de l'exponentiation modulaire

- Square and Multiply

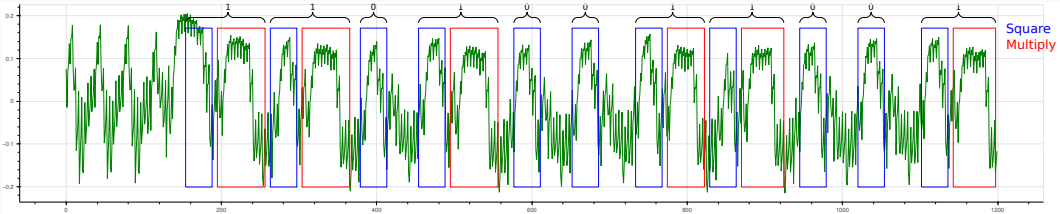


- Square and Multiply always

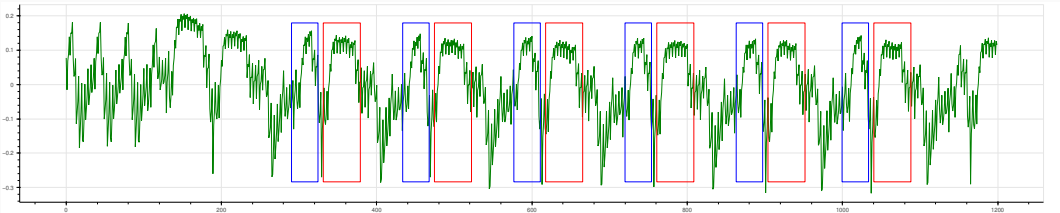


Simple Power Attacks : exemple de l'exponentiation modulaire

- Square and Multiply

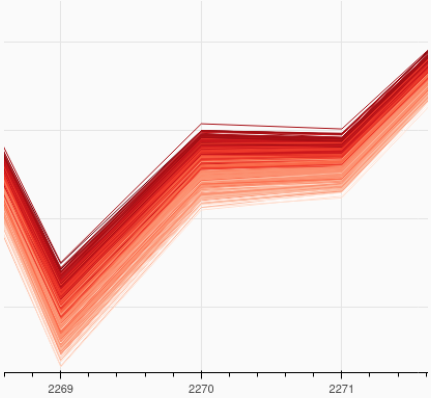


- Square and Multiply always



Variation de la consommation liée aux données

- En plus des instructions, les données aussi influencent la consommation
- Dans ce cas aussi le poids de Hamming peut être un bon modèle de consommation :

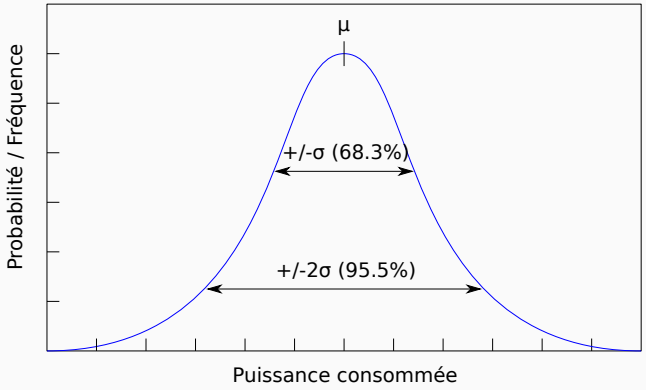


Bruit de mesure

- En réalité, deux mesures de consommation d'un même calcul vont être un peu différentes
- Il s'agit du Bruit électronique
- Raisons
  - Variations dans la tension d'alimentation
  - Variation du signal d'horloge
  - Émissions EM alentours
  - Modification de la température du système
  - Conversion analogique-numérique
- Il existe aussi un autre type de bruit quand on considère une attaque : le Bruit algorithmique (switching noise) : la consommation des cellules qui nous intéressent pour l'attaque n'est qu'une partie de la consommation mesurée
- Sur la carte ChipWhisperer Lite, on mesure la consommation globale de tout le processeur, le bruit algorithmique correspond donc à la consommation de toutes les cellules qui changent de valeur dans le(s) même(s) cycle(s) que les cellules attaquées
- En général pas gênant pour les SPA, doit être considéré pour les attaques différentielles : requiert un plus grand nombre de traces

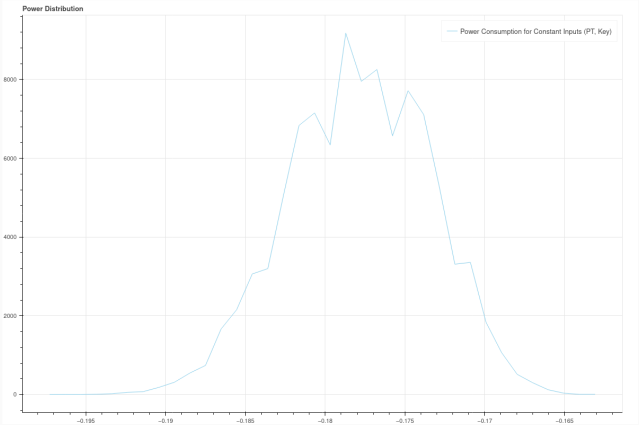
Caractérisation du bruit électronique

- La consommation en un point avec entrées fixées : peut être modélisée par une loi normale
- Somme de contributions/facteurs indépendants : loi binomiale
- Chaque facteur ajoute ou soustrait un petit peu de consommation



Caractérisation du bruit électronique

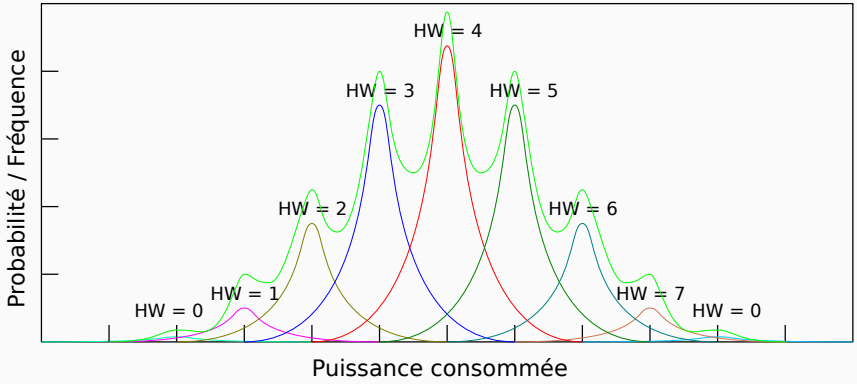
- D'après des mesures réelles (POI en sortie de la Sbox AES, key = 0xae, pt = 0xc4)



- Pas une loi normale : biais du capteur ? Problème de conversion ?

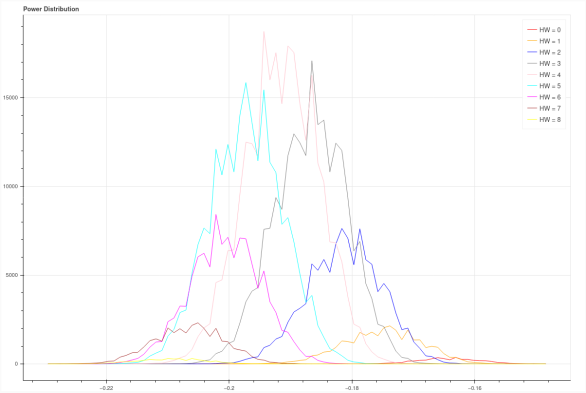
Influence des données sur la consommation

- Cas du chargement d'un octet, résultat de la SBox de l'AES
- Une distribution par valeur de poids de Hamming
- Distributions globale : somme de toutes ces distributions



Influence des données sur la consommation

- Cas du chargement d'un octet, résultat de la SBox de l'AES, mesures réelles



- Même effet/problème qu'avec les entrées fixées
- + Tous les bits ne consomment pas forcément la même quantité : un poids de Hamming identique peut mener à des consommations différentes si les bits à 1 ne sont pas les mêmes

Plan

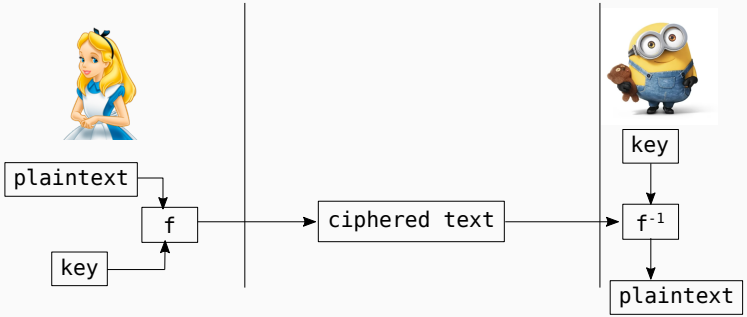
Introduction à l'UE

Systèmes cryptographiques et consommation électrique

Rappels sur le chiffrement symétrique et l'AES

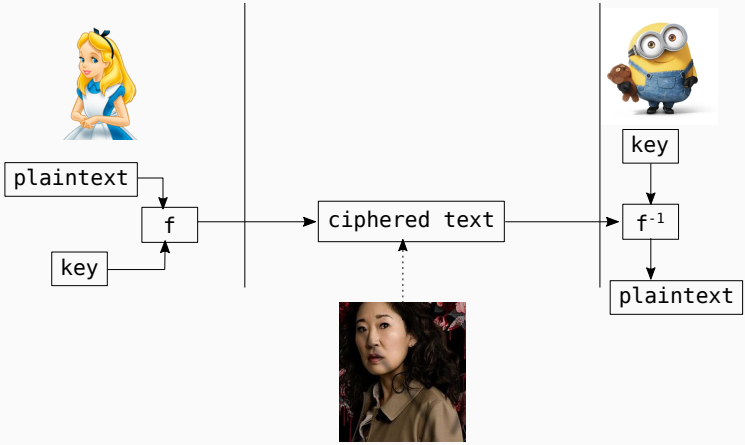
Chiffrement symétrique

- Algorithme de chiffrement symétrique :

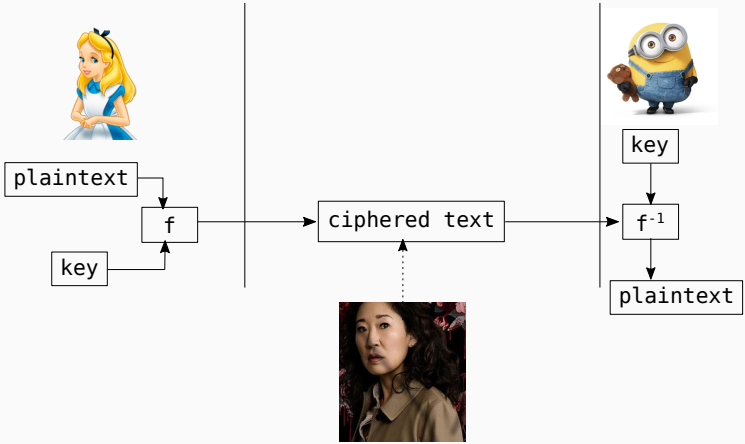


Chiffrement symétrique

- Algorithme de chiffrement symétrique :



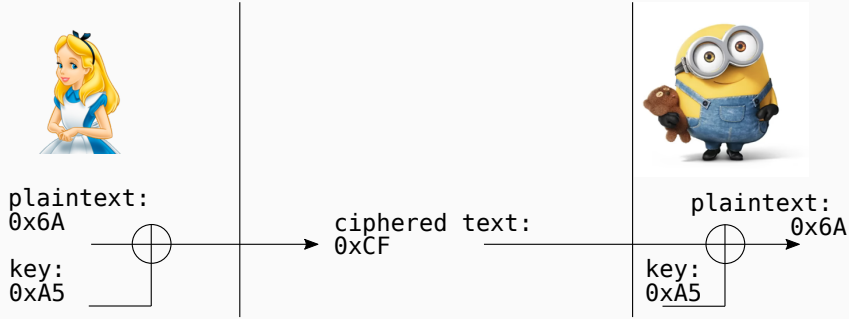
- Algorithme de chiffrement symétrique :



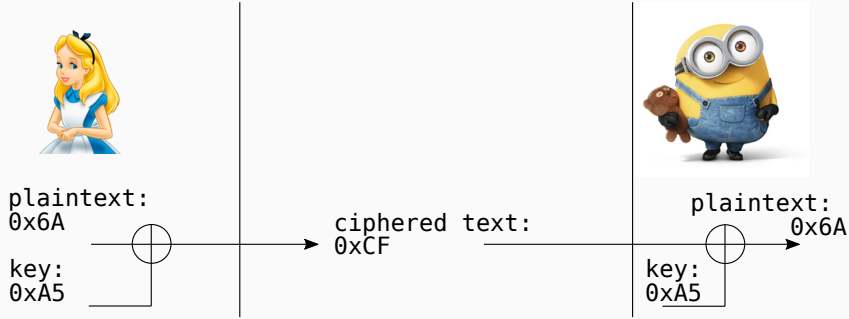
- Quelles propriétés et garanties ?

Chiffrement symétrique

- Pouquoi ne pas utiliser un $\oplus$ entre la clé (K) et le plaintext (PT) pour obtenir le chiffré (CT) ?



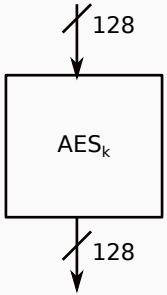
- Pouquoi ne pas utiliser un $\oplus$ entre la clé (K) et le plaintext (PT) pour obtenir le chiffré (CT) ?



- 1er Problème : on peut déduire des bits de clé si on connait des parties de PT (ex : protocole, en-tête, contrôle du dispositif)
  - Utiliser une clé aussi longue que le message et une nouvelle clé pour chaque message : infaisable en pratique
- 2e Problème : si l'attaquant a la possibilité de modifier le message : inverser un bit de CT revient à inverser un bit de PT

Rappels sur l'AES

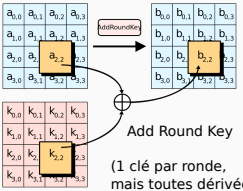
- Algorithme de chiffrement symétrique basé sur une clé secrète



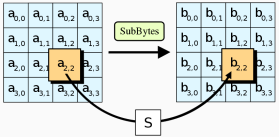
- Standard utilisé partout
- Plusieurs tailles de clés (128, 192 et 256 bits), version 128 la plus répandue
- Aucune faiblesse connue d'un point de vue cryptanalytique (cf. suite)
- ...Mais de nombreuses attaques utilisant le matériel ou les implémentations

Rappels sur l'AES : Structure

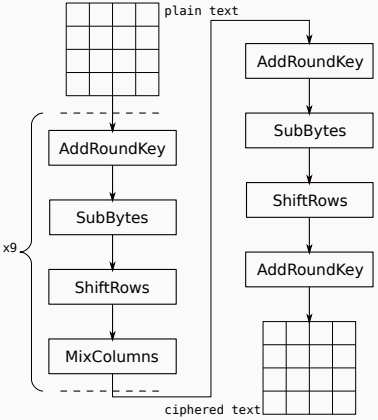
- 10 rondes (pour la version 128), avec les opérations AddRoundKey, SubBytes, ShiftRows, MixColumns (sauf dernière ronde)

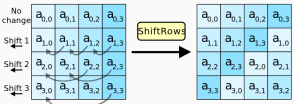


Source : Wikipedia

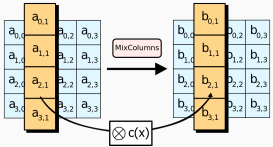


SubBytes





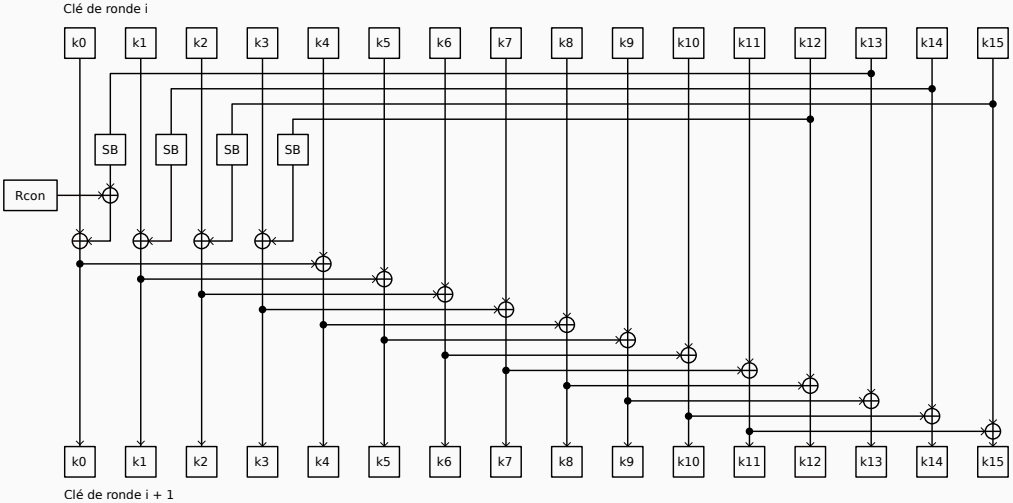
Shift Rows



Mix Columns

Rappels sur l'AES : Key Schedule

- Génération des 10 clés de ronde à partir de la clé originale



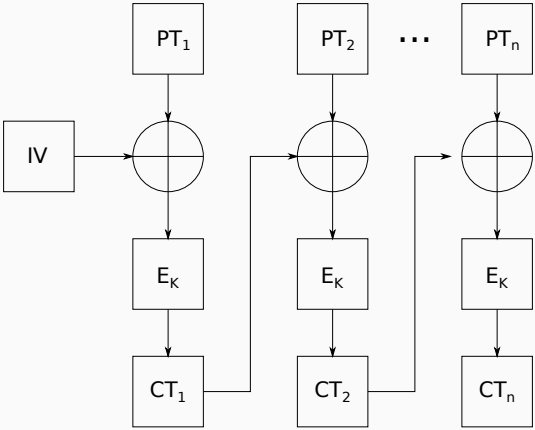
Chiffrement symétrique : propriétés de l'AES

- L'AES ne possède pas les deux problèmes vus précédemment :
  - 1. On peut observer autant de couples (PT, CT) que l'on veut (même avec PT ou CT choisi), cela ne permet pas de déduire de l'information sur la clé
  - 2. La modification d'un bit dans PT (resp. CT) résulte en la modification potentielle de tous les bits de CT (resp. PT) : propriété de diffusion
- Il reste un problème : le même PT chiffré deux fois donne deux fois le même CT
  - Logique... non ?
- Pour éviter cela, utilisation d'un vecteur d'initialisation (IV) : aléatoire (uniforme, indépendant, etc.)
- Alice fait CT = f(AES_k, IV, PT) puis envoie (IV, CT)
- La fonction f est telle que Bob peut déchiffrer le message en connaissant k, mais telle que Eve ne puisse rien déduire ni de PT ni de k à partir de (IV, CT)

- Dans le cas des messages "longs", i.e. de plus de 128 bits (toujours le cas en pratique), on évite de mettre un IV par bloc de message chiffré (1 bloc = 128 bits)
- $\Rightarrow$ On utilise un seul IV et on "chaine" les blocs entre eux
- Plusieurs modes de chainage : CBC, CFB, OFB
- Inconvénient : empêche de faire le chiffrement en parallèle des blocs...

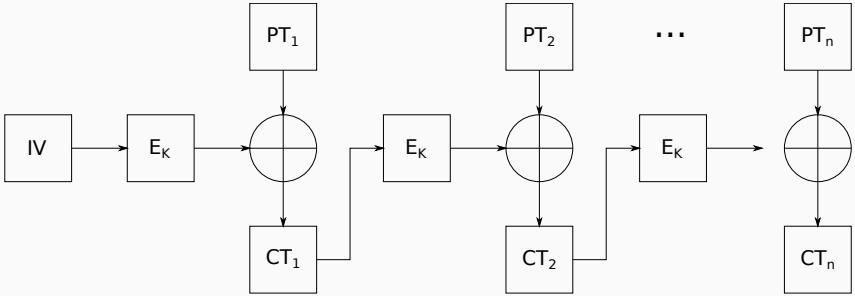
Modes de chainage de l'AES : Mode CBC

- Mode CBC : le plus utilisé

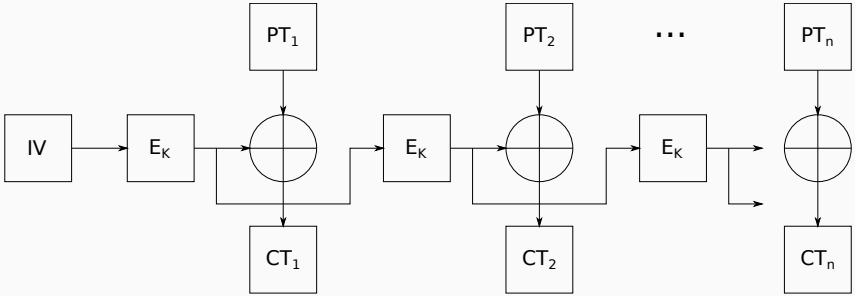


Modes de l'AES : Mode CFB

- Avantage : pas besoin de AES^{-1}

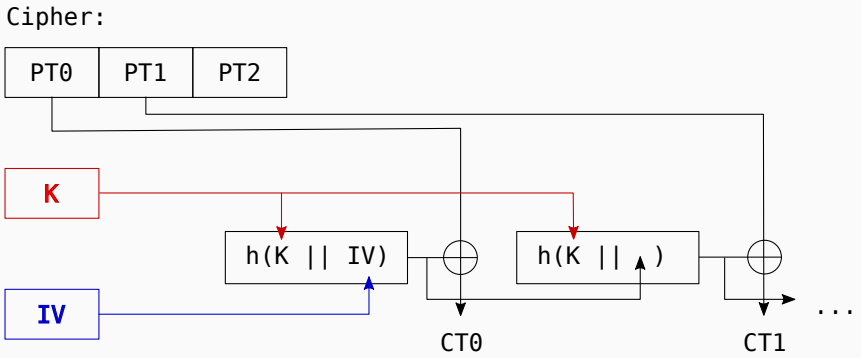


- Avantages : pas besoin de AES^{-1} et déchiffrement identique au chiffrement



Si l'on pousse le raisonnement plus loin...

- Seule une "bonne" fonction de hachage est nécessaire pour faire du chiffrement symétrique



- Déchiffrement : algorithme identique (inverser `PT` et `CT`)
- $\Rightarrow$ Plus besoin d'algorithme de chiffrement symétrique comme l'AES
- Confirmé par Joan Daemen en personne...

Merci !

Contact:
Email: quentin.meunier@lip6.fr