

## Opérateurs arithmétiques sécurisés

Arnaud Tisserand

IRISA, CNRS – Univ. Rennes 1  
Équipe-projet CAIRN

ARCHI 09  
Pleumeur-Bodou, 30 mars – 3 avril 2009



- Introduction  
sécurité = mathématiques + informatique + microélectronique
- Attaques physiques (exemples)  
analyse consommation/ rayonnement électromagnétique  
injection de fautes
- Opérateurs arithmétiques pour la crypto (exemples)  
opérations  $(\pm, \times, \frac{1}{x})$  entières modulo  $P$  sur 200–2000 bits
- Sécurisation d'opérateurs arithmétiques  
jouer avec les représentations des nombres, les algorithmes  
et les implantations
- Conclusion, perspectives, références

## Introduction

Les besoins en dispositifs de cryptographie sont de plus en plus grands :

- pour faire du chiffrement (ssh, site web sécurisé, diffusion de contenus via le réseau...)
- pour signer des documents électroniques
- pour authentifier des gens et des dispositifs
- pour vérifier l'intégrité d'un document
- pour assurer le non désaveu d'un document

Les besoins en intégration matérielle sont aussi de plus en plus grands :

- pour les performances (vitesse, taille/poids, consommation)
- pour la sécurité (éviter les attaques)

## Terminologie

Cryptographie = art de garder les messages secrets

Cryptanalyse = art de décrypter les messages chiffrés

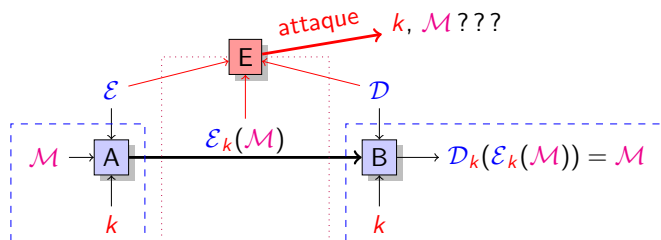
Cryptologie = cryptographie + cryptanalyse

Stéganographie = art de cacher les messages dans d'autres messages

Attaque = tentative pour, sans connaître le secret, retrouver :

- des messages (ou des bouts de message)
- des informations sur le message
- le secret (ou des bouts du secret)

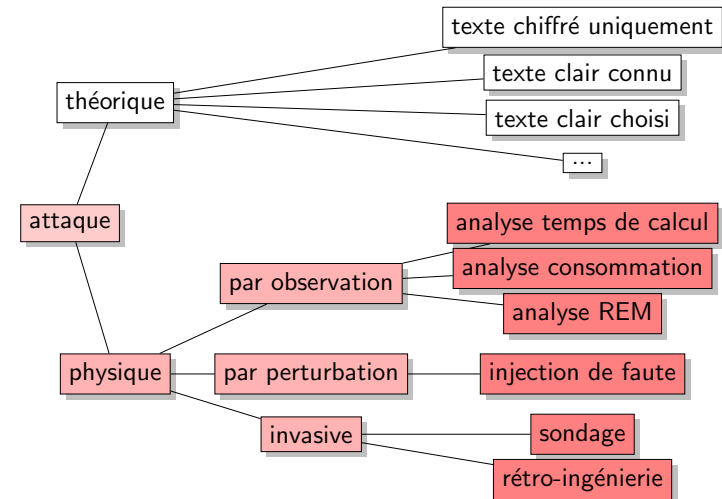
## Exemple : chiffrement en crypto symétrique



### Notations :

- $\mathcal{M}$  message en clair
- $\mathcal{E}$  algorithme de chiffrement
- $\mathcal{D}$  algorithme de déchiffrement
- $k$  clé secrète
- $\mathcal{C} = \mathcal{E}_k(\mathcal{M})$  message chiffré
- $\boxed{\phantom{x}}$  zone sécurisé
- $\boxed{\phantom{x}}$  canal de communication

## Quelques types d'attaques



REM = rayonnement électromagnétique

## Attaques théoriques

Tout à fait réalisable en pratique avec :

- maths
- algorithmique de compétition
- programmation hyper hyper optimisée

Exemple : factorisation de grands entiers pour casser RSA

- RSA 512 bits, août 1999
- RSA 576 bits, décembre 2003
- RSA 640 bits, novembre 2005, équipe allemande avec 80  $\mu$ P Opteron à 2.2 GHz
- RSA 768 bits en cours...

## Pour éviter les attaques théoriques

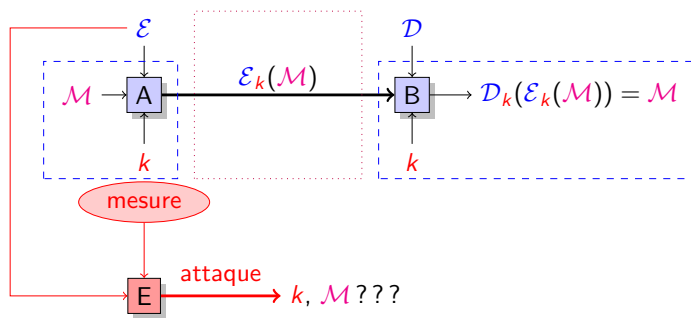
Suivre les recommandations des spécialistes...

Exemple : courbe elliptique P-521 sur un corps premier recommandée par le NIST (cf. FIPS 186-2)

```
p = 68647976601306097149819007990813932172694353
00143305409394463459185543183397656052122559
64066145455497729631139148085803712198799971
6643812574028291115057151
r = 68647976601306097149819007990813932172694353
00143305409394463459185543183397655394245057
74633321719753296399637136332111386476861244
0380340372808892707005449
s = d09e8800 291cb853 96cc6717 393284aa a0da64ba
c = 0b4 8bfa5f42
0a349495 39d2bdfc 264eeeb 077688e4 4fbf0ad8
f6d0edb3 7bd6b533 28100051 8e19f1b9 ffbef0e9
ed8a3c22 00b8f875 e523868c 70c1e5bf 55bad637
.....
```

## Attaque physique par canaux cachés

En anglais : *Side Channel Analysis/Attacks (SCA)*



**Principe :** mesurer des **paramètres externes** du dispositif pendant son fonctionnement pour en déduire des **informations internes**

## Que mesurer ?

**Réponse :** **tout** ce qui peut “entrer” et/ou “sortir” dans le/du dispositif

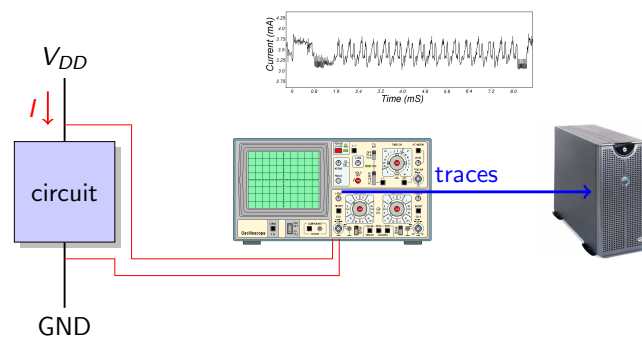
- la consommation d'énergie
- le rayonnement électromagnétique
- la température
- le bruit (son)
- le temps de calcul
- le nombre de défauts de cache
- le nombre et le type de messages d'erreur
- ...

Ce que l'on mesure peut donner des informations sur le comportement :

- **global** (température, consommation, bruit...)
- **local** (REM, nb défauts de cache...)

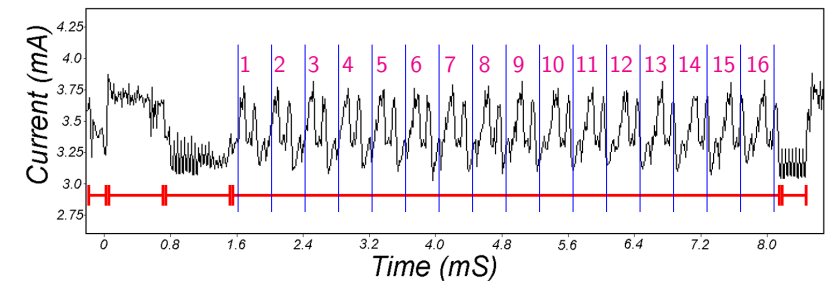
## Attaque par mesure de la consommation d'énergie

**Principe :** mesurer le courant  $I$  qui alimente le dispositif



**Notations :**  $V_{DD}$  tension d'alimentation (5, 3, 2.5, 1.2 V), GND la masse

## Que lire dans les traces ?



- algorithme  $\Rightarrow$  découpage en étapes
- détection des tours de boucle (calculs répétitifs)
  - ▶ temps constant dans un tour
  - ▶ ou pas ???

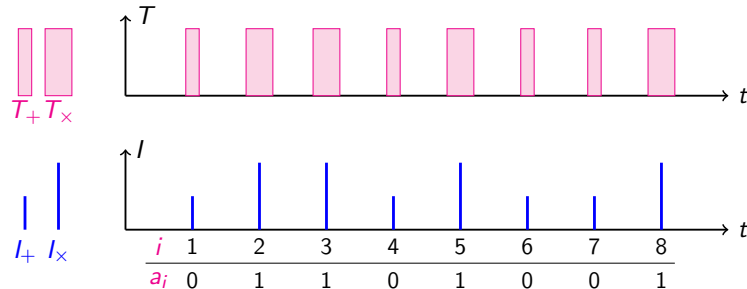
## Exploiter les différences

Un algorithme a une **signature** en **courant** et en **temps de calcul** :

```

r = c0
for i from 1 to n do
  if ai = 0 then
    r = r + c1
  else
    r = r × c2

```

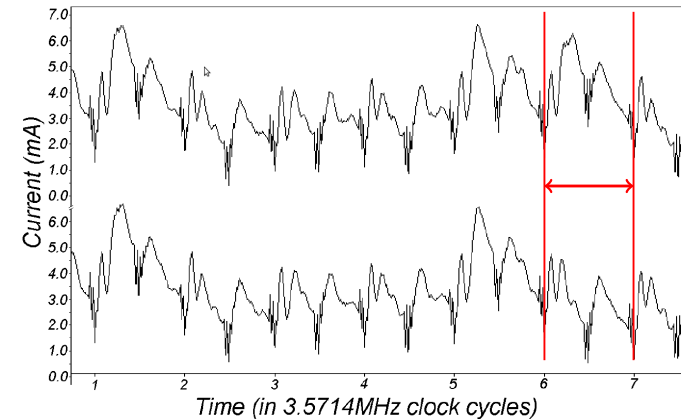


A. Tisserand, IRISA, CNRS-Univ. Rennes 1. Opérateurs arithmétiques sécurisés

13/54

## Analyse simple de la consommation (SPA)

En anglais : SPA *simple power analysis*

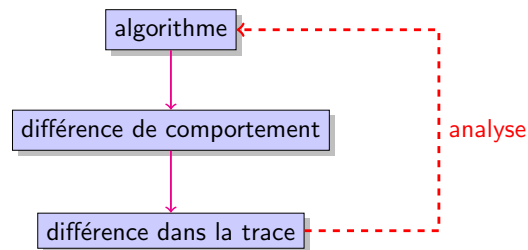


A. Tisserand, IRISA, CNRS-Univ. Rennes 1. Opérateurs arithmétiques sécurisés

14/54

## La SPA en pratique

Principe :

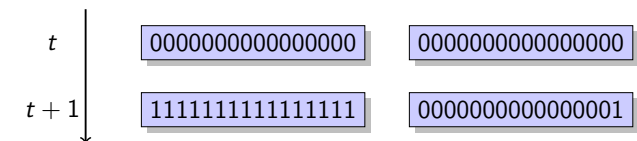


**Techniques** : exploiter les différences dans

- le contrôle
- le temps de calcul
- les valeurs des opérandes (temps de calcul, conso., REM...)
- ...

## Limites de la SPA

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

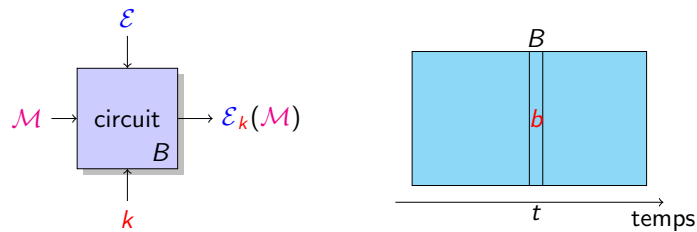


**Important** : une petite variation de comportement peut être plus ou moins cachée par du **bruit**  $\Rightarrow$  les traces ne sont plus discernables

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

**Réponse** : utiliser des **statistiques** sur des **nombreuses** courbes

## État interne d'un cryptosystème



### Notations :

- $t$  un instant de l'exécution ( $t \in \{1, \dots, T\}$ )
- $B = F_{\mathcal{E}}(\mathcal{M}, k, t)$  état interne du cryptosystème
- **IMPORTANT** :  $B$  n'est pas accessible de l'extérieur !

**Objectif** : essayer de découvrir  $b$  un élément de  $B$  (ex : un bit)

## Analyse différentielle de la consommation (DPA) (1/2)

En anglais : DPA *differential power analysis*

### Principe :

1. effectuer  $N$  exécutions du cryptosystème
  - ▶ on garde les messages en clair  $\mathcal{M}_i$  ( $i \in \{1, \dots, N\}$ )
  - ▶ on mesure les traces  $P_{ij}$  ( $j \in \{1, \dots, T\}$ )
2. calculer la trace moyenne  $\bar{P}_j = \frac{1}{N} \sum_{i=1}^N P_{ij}$
3. sélectionner un bit  $b$  à attaquer (c.a.d. trouver  $b$  en interne)
4. partitionner les traces  $P_{ij}$  en deux ensembles :
  - ▶  $S_0$  celles qui correspondent à  $b = 0$  (tous les  $i$  qui donnent  $b = 0$ )
  - ▶  $S_1$  celles qui correspondent à  $b = 1$  (tous les  $i$  qui donnent  $b = 1$ )
5. faire une hypothèse sur  $b$  :  $H = H_{b=0}$  ou  $H_{b=1}$
6. comparer statistiquement la trace moyenne globale  $\bar{P}_j$  à la trace moyenne de  $S_0$  ou  $S_1$  (celle de  $H$ )

## Analyse différentielle de la consommation (DPA) (2/2)

Supposons  $H = H_{b=0}$  on compare  $\bar{P}_j$  et la moyenne des traces de  $S_0$

La comparaison peut donner :

- aucune différence significative mesurable  $\Rightarrow H$  était **incorrecte** (c.a.d.  $b \neq 0$ )
- une différence est mesurable à l'instant  $t \Rightarrow H$  était **correcte** (c.a.d.  $b = 0$ )

**Remarque** : idem avec l'autre hypothèse

Supposons  $H = H_{b=1}$  on compare  $\bar{P}_j$  et la moyenne des traces de  $S_1$

La comparaison peut donner :

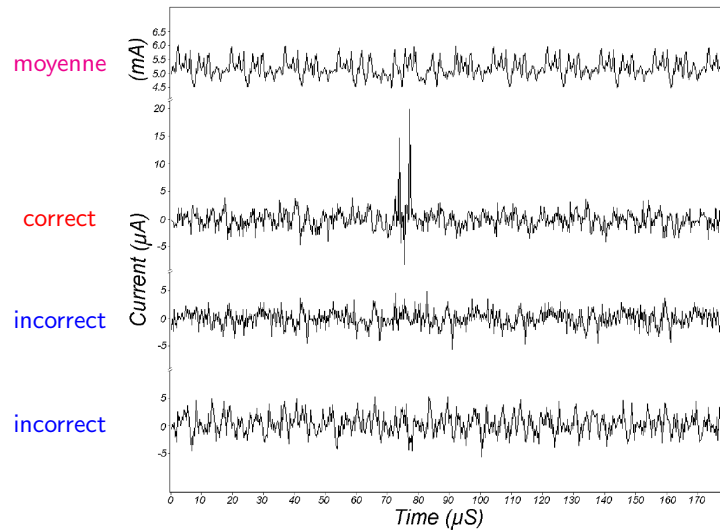
- aucune différence significative mesurable  $\Rightarrow H$  était **incorrecte** (c.a.d.  $b \neq 1$ )
- une différence est mesurable à l'instant  $t \Rightarrow H$  était **correcte** (c.a.d.  $b = 1$ )

## Pourquoi la DPA marche ?

**Réponse** : grâce au partitionnement  $S_0 / S_1$  par rapport à  $H$

- si l'hypothèse  $H$  est **incorrecte**
  - $\Rightarrow$  les  $N$  exécutions/traces correspondent à un  $b$  faux
  - $\Rightarrow$  le partitionnement  $S_0 / S_1$  est **aléatoire**
  - $\Rightarrow$  si  $N$  est **grand**, la différence entre la moyenne globale et la moyenne de l'ensemble testé sont **proches** à l'instant  $j = t$
- si l'hypothèse  $H$  est **correcte** :
  - $\Rightarrow$  les  $N$  exécutions/traces correspondent à un  $b$  correct
  - $\Rightarrow$  le partitionnement  $S_0 / S_1$  est **significatif**
  - $\Rightarrow$  si  $N$  est **grand**, la différence entre la moyenne globale et la moyenne de l'ensemble testé sont **différentes** à l'instant  $j = t$  car on **exploite la différence de comportement** entre  $b = 0$  et  $b = 1$

## Exemple de courbes obtenues par DPA

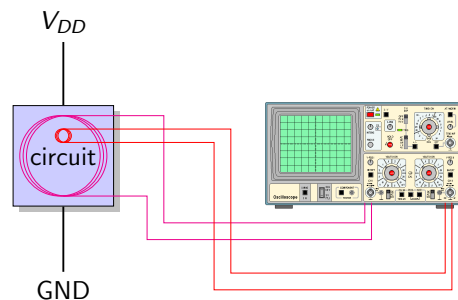


## Remarques sur la DPA

- partitionner demande de connaître le  $b$  théorique pour chaque message testé  $\mathcal{M}_i$ ;
- il faut que le  $N$  soit assez grand pour deux raisons :
  - ▶ amplifier la différence dans le cas de  $H$  correcte
  - ▶ rendre aléatoire la différence dans le cas de  $H$  incorrecte
- connaître  $t$  n'est pas nécessaire pour faire l'attaque, mais ça aide à restreindre la taille des traces (donc le temps d'attaque)
- ce qui est **difficile** c'est de savoir quel(s)  $b$  attaquer !
  - ▶ il faut que  $b$  engendre une différence de comportement
  - ▶  $b$  peut être un bit ou **plusieurs** bits
- on utilise des tests statistiques de plus en plus sophistiqués
- attaque particulièrement efficace en pratique

## Analyse du rayonnement électromagnétique (1/2)

**Principe** : utiliser une **sonde** qui va capter le REM.



**Mesure du REM** :

- **global** avec une **grande sonde**
- **local** avec une **microseconde**

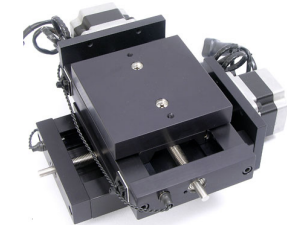
## Analyse du rayonnement électromagnétique (2/2)

Types d'analyse du REM :

- **simple** : SEMA (*simple electromagnetic analysis*)
- **différentielle** : DEMA (*differential electromagnetic analysis*)

Le caractère **local** de l'analyse du REM peut être vraiment intéressant pour essayer de déterminer l'architecture du circuit, puis d'attaquer des endroits bien précis.

⇒ table X-Y



## Attaques par injection de fautes (1/2)

**Principe** : essayer de **faire varier** certains paramètres du dispositif par rapports à ses **conditions normales** de fonctionnement en espérant que cela engendrera une différence de comportement interne, une **faute**, mesurable à l'extérieur

### Comment perturber ?

- modifier la tension d'alimentation
- modifier la température
- modifier le signal d'horloge
- ne pas respecter les temps de *setup/hold* sur les entrées
- soumettre le circuit à un REM
- illuminer le circuit avec un laser
- ...

## Attaques par injection de fautes (2/2)

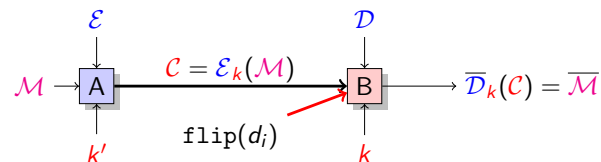
### Quelques types de fautes :

- coller des valeurs internes à 1 (ou 0)
- inverser la valeur d'un bit (*flip*)
- interdire certaines transitions (ex :  $0 \rightarrow 1$  OK mais pas  $1 \rightarrow 0$ )
- empêcher un saut dans un branchement
- modifier le décodage des instructions
- ...

Ensuite on utilise les fautes pour créer des différences de comportement et essayer de mesurer ces différences en externe

### Exemple d'attaque en faute

Attaque du déchiffrement RSA par inversion de bit (*flip*)



- choisir un message en clair  $\mathcal{M}$
- chiffrer  $\mathcal{M}$  en  $\mathcal{C} = \mathcal{E}_k(\mathcal{M})$
- injecter une faute en inversant  $d_i$  avec  $i$  aléatoire ( $d$  clé secrète)
- calculer  $\overline{\mathcal{M}} = \frac{c^{2^i d_i}}{c^{2^i d_i}}$
- test :
  - ▶  $\frac{\overline{\mathcal{M}}}{\mathcal{M}} = \frac{1}{c^{2^i}} \bmod N \Rightarrow d_i = 1$
  - ▶  $\frac{\overline{\mathcal{M}}}{\mathcal{M}} = c^{2^i} \bmod N \Rightarrow d_i = 0$
- recommencer avec plusieurs  $i$  ( $\Rightarrow$  bouts de  $d$ , puis attaque maths)

### Contre-mesures contre les attaques

**Empêcher** une (ou des) attaques par :

- un nouveau dispositif de protection
- la modification/**sécurisation** du dispositif original

### Exemples :

- blindage
- uniformiser les temps de calcul
- uniformiser la consommation d'énergie
- utiliser des codes détecteurs/correcteurs d'erreurs
- introduire du bruit (instructions inutiles)
- reconfigurer le circuit
  - ▶ changer le codage des données
  - ▶ changer les algorithmes

## Opérateurs arithmétiques pour la crypto

Éléments de :

- corps fini premier  $\text{GF}(p)$  ( $p$  un grand nombre premier)
- extensions du corps binaire  $\text{GF}(2^m)$
- extensions de corps premiers  $\text{GF}(p^m)$  (ex :  $p = 3$ )

Tailles classiques en crypto à clé publique :

- RSA  $\Rightarrow$  1024 à 8192 bits
- ECC  $\Rightarrow$  200 à 600 bits

**Bonne référence :** *Guide to ECC* (cf. livres)

**Notation :** ECC *elliptic curve cryptography*

## Addition modulo $M$

Entrées :

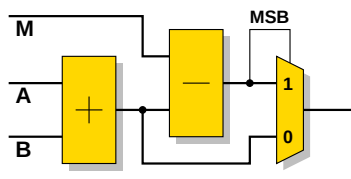
$$A, B \in \{0, 1, 2, 3, \dots, M-1\}$$

Sortie<sup>1</sup> :

$$(A + B) \bmod M$$

Algorithme :

$$(A+B) \bmod M = \begin{cases} A+B & \text{si } A+B < M \\ A+B-M & \text{si } A+B \geq M \end{cases}$$



<sup>1</sup> $0 \leq A + B \leq 2M - 2$

Corps fini premier  $\text{GF}(p)$

- $p$  est un **grand** nombre **premier**
- éléments du corps :  $\{0, 1, 2, 3, \dots, p-1\}$
- dans le corps, calcul sur les entiers **modulo**  $p$

Exemple : GF(29)

- éléments :  $\{0, 1, 2, 3, \dots, 27, 28\}$

- opérations modulo 29

addition :  $17 + 20 = 8 \text{ car}$        $37 \bmod 29 = 8$

soustraction :  $17 - 20 = 26$  car  $-3 \bmod 29 = 26$

multiplication :  $17 \times 20 = 21$  car  $340 \bmod 29 = 21$

inversion :  $17^{-1} = 12$  car  $17 \times 12 \bmod 29 = 1$

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|

Addition modulo  $2^n - 1$

Entrées :

$$A, B \in \{0, 1, 2, 3, \dots, 2^n - 2\}$$

Sortie :

$$(A + B) \bmod (2^n - 1)$$

Méthode naïve :

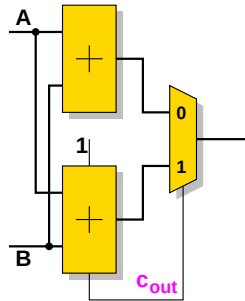
$$(A+B) \bmod (2^n - 1) = \begin{cases} A+B & \text{si } A+B < 2^n - 1 \\ \underbrace{A+B - (2^n - 1)}_{A+B+1} & \text{si } A+B \geq 2^n - 1 \end{cases}$$

**Problème** : la condition  $A + B \geq 2^n - 1$  est coûteuse



## Addition modulo $2^n - 1$ : variante 1

$$(A + B) \bmod (2^n - 1) = \begin{cases} A + B & \text{si } A + B + 1 < 2^n \\ A + B + 1 & \text{si } A + B + 1 \geq 2^n \end{cases}$$



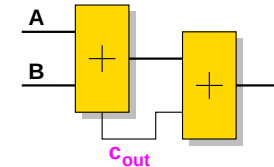
## Addition modulo $2^n - 1$ : variante 2

**Idée** : double représentation de 0 :  $00 \dots 00 = 11 \dots 11$

$$(A + B) \bmod (2^n - 1) \in \{0, 1, 2, 3, \dots, 2^n - 2, 2^n - 1\}$$

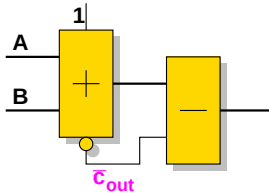
Méthode modifiée :

$$(A + B) \bmod (2^n - 1) = \begin{cases} A + B & \text{si } A + B < 2^n \\ A + B + 1 & \text{si } A + B \geq 2^n \end{cases} \\ = A + B + c_{out}$$



## Addition modulo $2^n - 1$ : dernière variante

$$(A + B) \bmod (2^n - 1) = (A + B + 1) \bmod 2^n + \bar{c}_{out}(2^n - 1) \\ = (A + B + 1) \bmod 2^n - \bar{c}_{out}$$



**Optimisation** : travailler au niveau arithmétique ET et au niveau architecture/circuit

## Addition modulo

Techniques similaires pour l'addition modulo  $2^n + 1$  ou des moduli spécifiques

**Mais** :

- dépend de la valeur (écriture au niveau chiffres) de  $M$
- difficile d'optimiser pour plusieurs moduli ( $M_1, M_2, \dots, M_k$ )
- dépend aussi de la cible

**Opérations modulaires utiles** :

- $A + B + 1$  et  $A + B - 1$
- $A + B$  et  $A + B + 1$  en même temps

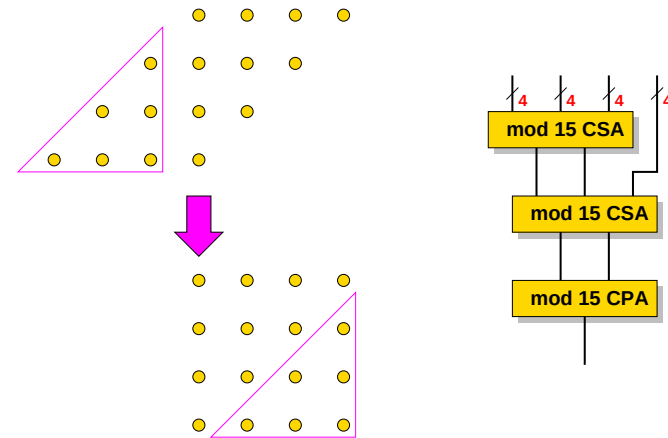
## Multiplication modulaire

Deux méthodes :

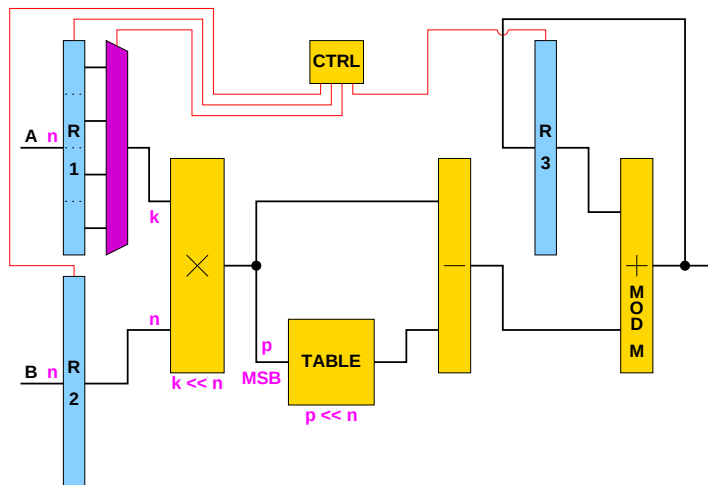
- multiplication **et** réduction modulaire
  - ▶ **étape 1** : calculer  $P = A \times B$
  - ▶ **étape 2** : réduire  $R = P \bmod M$
  - ▶ possible pour des  $M$  comme  $2^n - 1$ ,  $2^n + 1$  ou d'autres moduli spécifiques (mais pas en général)
- accumulation (modulaire) des **produits partiels réduits**
  - ▶ parallèle-parallèle
  - ▶ série-parallèle (MSB ou LSB en premier)

## Multiplication modulaire : $(A \times B) \bmod 15$

**Principe** :  $16 \bmod 15 = 1$



## Multiplication modulaire : compromis



## Exponentiation modulaire

**Algorithme** : *square and multiply*

Entrées :  $x$ ,  $d = (d_{m-1} \dots d_1 d_0)_2$   
Sortie :  $y = x^d$

```

1   $R \leftarrow 1$ 
2   $i \leftarrow m - 1$ 
3  while ( $i \geq 0$ ) do
4       $R \leftarrow R^2$  square
5      if ( $d_i = 1$ ) then
6           $R \leftarrow R \times x$  multiply
7      endif
8       $i \leftarrow i - 1$ 
9  endwhile
10 return  $R$ 
    
```

Opération majeure dans RSA

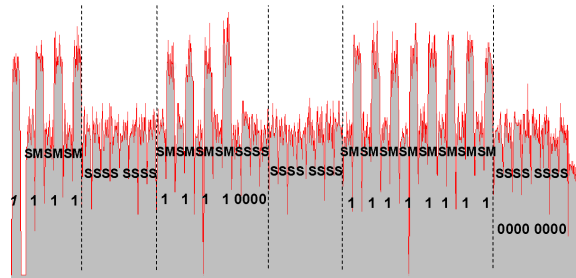
## Square and multiply : vulnérable !

### Attaque : SPA

Différence de comportement à chaque tour de boucle :

- $d_i = 1 \Rightarrow$  carré et multiplication
- $d_i = 0 \Rightarrow$  carré

Exemple de trace :



## Contre-mesure SPA : Square and multiply always

Entrées :  $x, d = (d_{m-1} \dots d_1 d_0)_2$   
Sortie :  $y = x^d$

```

1   $R \leftarrow 1$ 
2   $i \leftarrow m - 1$ 
3  while ( $i \geq 0$ ) do
4       $R_1 \leftarrow R^2$                                 square
5       $R_2 \leftarrow R_1 \times x$                         multiply
6      if ( $d_i = 1$ ) then
7           $R \leftarrow R_2$ 
8      else
9           $R \leftarrow R_1$ 
10     endif
11      $i \leftarrow i - 1$ 
12 endwhile
13 return  $R$ 

```

## ECC : multiplication scalaire

C'est l'opération principale

Entrée :  $P$  un point de la courbe  $E$ , un grand entier  $k = \sum_{i=0}^{n-1} k_i 2^i$

Sortie : le point  $Q = [k]P = \underbrace{P + P + P + \dots + P}_{k \text{ fois}}$

Algorithme classique : *double-and-add*

```

1 :  $Q \leftarrow P$ 
2 : for  $i$  from  $n-2$  to  $0$  do
3 :    $Q \leftarrow 2P$ 
4 :   if  $k_i = 1$  then  $Q \leftarrow Q + P$ 

```

Problème : pas robuste à la SPA !

## Recoder la clé $k$ pour être plus résistant

Recodages  $w$ -NAF (*non-adjacent form*)

Dans

$$k = \sum_{i=0}^{n-1} k_i 2^i, \quad k_i \in \{0, 1\}$$

parcourir  $k$  par paquets de  $w$  chiffres

$$|k_i| < 2^{w-1}$$

Exemple :

$$\begin{aligned}
 k = 267 &= ( \begin{array}{cccccccc} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{array} )_2 \\
 &= ( \begin{array}{cccccccc} 1 & 0 & 0 & 0 & 1 & 0 & \bar{1} & 0 & \bar{1} \end{array} )_{2\text{-NAF}} \\
 &= ( \begin{array}{cccccccc} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 3 \end{array} )_{3\text{-NAF}} \\
 &= ( \begin{array}{cccccccc} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \bar{5} \end{array} )_{4\text{-NAF}} \\
 &= ( \begin{array}{cccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 11 \end{array} )_{5\text{-NAF}}
 \end{aligned}$$

Coût :  $n - 1$  DBL et  $\frac{n}{w+1}$  ADD

## Chaîne d'additions (thèse Nicolas Meloni)

Utiliser uniquement des additions :

- résistant à la SPA
- à chaque étape  $ADD(P_1, P_2) = (P_1 + P_2, P_1)$  avec  $P_1$  et  $P_2$  connus
- mais il faut trouver une chaîne courte

Exemple :  $k = 34$

|    |   |    |   |    |
|----|---|----|---|----|
| 34 | - | 19 | = | 15 |
| 19 | - | 15 | = | 4  |
| 15 | - | 4  | = | 11 |
| 11 | - | 4  | = | 7  |
| 7  | - | 4  | = | 3  |
| 4  | - | 3  | = | 1  |
| 3  | - | 1  | = | 2  |
| 2  | - | 1  | = | 1  |
| 1  | - | 1  | = | 0  |

## Double-Base Number Systems (DBNS) (1/3)

Source : L. Imbert

Représentation **redondante** basée sur la somme de puissances de 2 et 3 :

$$x = \sum_{i=1}^n x_i 2^{a_i} 3^{b_i}, \text{ avec } x_i \in \{-1, 1\}, a_i, b_i \geq 0$$

Exemple :  $127 = 108 + 16 + 3 = 72 + 54 + 1 = \dots$

|    |   |   |   |   |    |
|----|---|---|---|---|----|
|    | 1 | 2 | 4 | 8 | 16 |
| 1  |   |   |   |   | 1  |
| 3  | 1 |   |   |   |    |
| 9  |   |   |   |   |    |
| 27 |   |   | 1 |   |    |

|    |   |   |   |   |
|----|---|---|---|---|
|    | 1 | 2 | 4 | 8 |
| 1  | 1 |   |   |   |
| 3  |   |   |   |   |
| 9  |   |   |   | 1 |
| 27 |   | 1 |   |   |

## Double-Base Number Systems (DBNS) (2/3)

Plus petit  $x > 0$  nécessitant  $n$  termes DBNS :

| $n$ | non-signé     | signé  |
|-----|---------------|--------|
| 2   | 5             | 5      |
| 3   | 23            | 105    |
| 4   | 431           | (4985) |
| 5   | 18,431        | ?      |
| 6   | 3,448,733     |        |
| 7   | 1,441,896,119 |        |
| 8   | ?             |        |

Exemple : 127 a exactement 783 représentations DBNS, dont 6 sont canoniques :  $127 = (108 + 18 + 1) = (108 + 16 + 3) = (96 + 27 + 4) = (72 + 54 + 1) = (64 + 54 + 9) = (64 + 36 + 27)$

## Double-Base Number Systems (DBNS) (3/3)

Application : multiplication scalaire ECC

$$314159 = 2^4 3^9 + 2^8 3^1 - 1$$

$$[314159]P = [2^4 3^9]P + [2^8 3^1]P - P$$

coût : 12 DBL + 10 TPL + 2 ADD

$$314159 = 2^4 3^9 - 2^0 3^6 - 3^3 - 3^2 - 3 - 1$$

$$[314159]P = 3(3(3(3^3([2^4 3^3]P - P) - P) - P) - P) - P$$

coût : 4 DBL + 9 TPL + 5 ADD

## Représentations des chiffres au niveau circuit

Représentation classique d'un bit  $b$  :

- $V_{DD} \Rightarrow b = 1$
- $GND \Rightarrow b = 0$

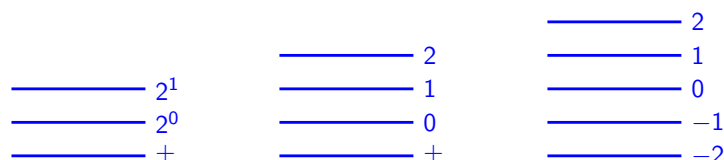
Représentation **double-rail** d'un bit  $b$  :

- $r_1 = V_{DD} \ r_0 = GND \Rightarrow b = 1$
- $r_1 = GND \ r_0 = V_{DD} \Rightarrow b = 0$

**Intérêt** : le nombre de transitions sur les fils (donc l'activité) est le même pour les transitions logiques  $0 \rightarrow 1$  et  $1 \rightarrow 0$

**Coût** : en surface de circuit et en mémoire

**Codages en grande base** : base 4 avec les chiffres  $\{-2, -1, 0, 1, 2\}$



## Conclusion & perspectives

- **attaques** de plus en plus **performantes**
- sécurisation nécessaire à **tous les niveaux** (algo, opération, implantation)
- sécurisation = compromis performances / robustesse
- coût de sécurisation =  $f(\text{valeur secret, type attaquant})$
- **sécurité** = mathématiques + informatique + microélectronique

**Exemple de recherches actuelles** :

- exploiter les représentations redondantes
- reconfiguration du circuit (représentations, algos)
- rendre l'activité moins dépendante des valeurs
- représentation des nombres avec code détecteur/correcteur d'erreur
- liens ordonnancement des sous-calculs avec l'activité du circuit
- exploration des compromis performances/robustesse

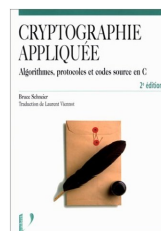
## Livres (1/2)

### Cryptographie appliquée (2<sup>e</sup> édition)

B. Schneier

2001. Vuibert

ISBN : 2-7117-8676-5

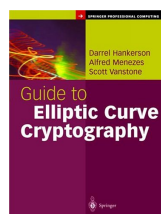


### Guide to Elliptic Curve Cryptography

D. Hankerson, A. Menezes and S. Vanstone

2004. Springer

ISBN : 0-387-95273-X



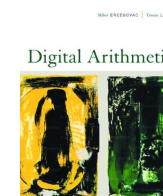
## Livres (2/2)

### Digital Arithmetic

Milos Ercegovac and Tomas Lang

2003. Morgan Kaufmann

ISBN : 1-55860-798-6

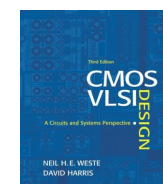


### CMOS VLSI Design (3rd edition)

N. Weste and D. Harris

2004. Addison Wesley

ISBN : 0-321-14901-7



## Quelques références

- *Differential Power Analysis*. Paul Kocher, Joshua Jaffe, and Benjamin Jun. Whitepaper from Cryptography Research, Inc.  
<http://www.cryptography.com>
- *A Tutorial on Physical Security and Side-Channel Attacks*. François Koeune and François-Xavier Standaert.  
<http://www.dice.ucl.ac.be/crypto/>
- *The Sorcerer's Apprentice Guide to Fault Attacks*. Hagai Bar-El, Hamid Choukri, David Naccache, Michael Tunstall and Claire Whelan <http://citeseer.ist.psu.edu/old/694897.html>
- [http://www.crypto.ruhr-uni-bochum.de/en\\_sclounge.html](http://www.crypto.ruhr-uni-bochum.de/en_sclounge.html)
- <http://www.schneier.com/>

## Fin, des questions ?

### Contact:

- <mailto:arnaud.tisserand@irisa.fr>
- <http://www.irisa.fr/prive/Arnaud.Tisserand/>
- Equipe-projet CAIRN
- Laboratoire IRISA, CNRS–Univ. Rennes 1  
6 rue Kérampont, BP 80518, F-22305 Lannion cedex, France

Merci