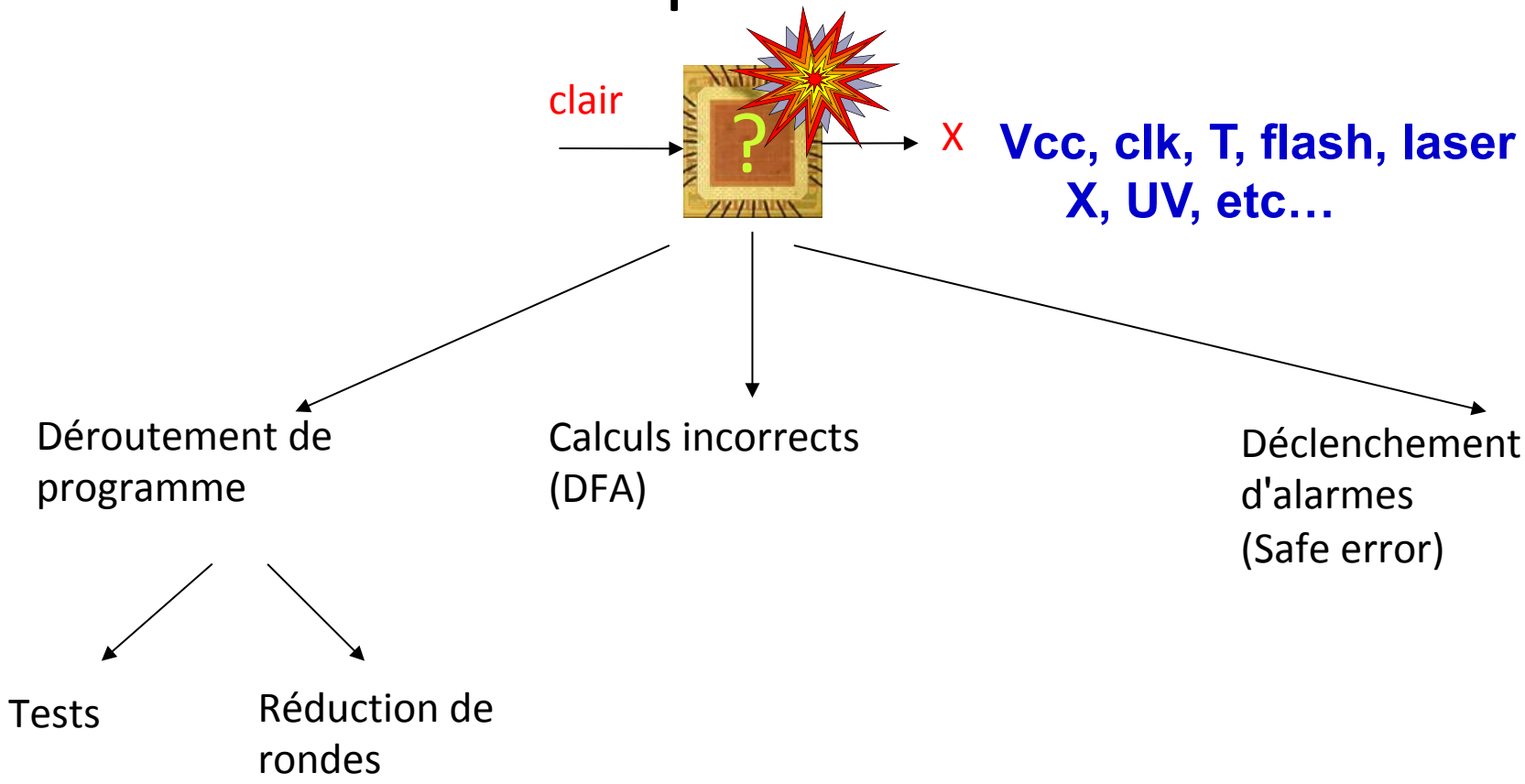
Attaques en fautes

Attaques en faute



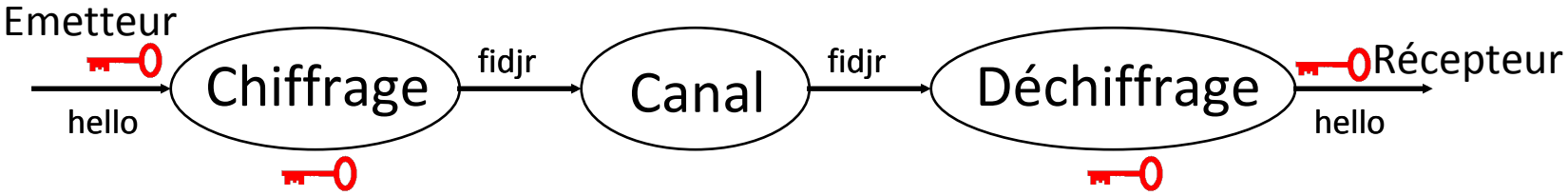
Syndrome :

Le fonctionnement du circuit peut être perturbé par la modification de son environnement

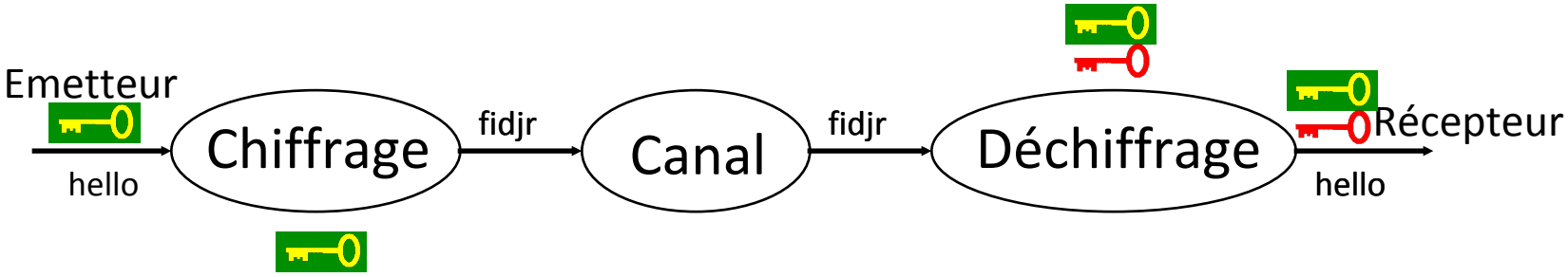
Attaque sur le RSA

Rappel : RSA

- Cryptographie à clé « privée » cryptage symétrique
  - Clé petite (128 bits), chiffrement rapide



- Cryptographie à clé « publique » cryptage asymétrique
  - 2 grandes clés (512 à 1024 bits) :
    - 1 publique connue de tous
    - 1 privée pour déchiffrer



Rappel

Théorème de Fermat :

si p premier, alors $a^p = a \pmod p$

Identité de Bezout

soit a et b relatifs, il existe u et v / $a.u + b.v = pgcd(a,b)$

Conséquence 1 :

si a et b premiers entre eux : $a.u + b.v = 1$

autrement dit : $a.u = 1 \pmod b$

u : inverse modulaire de a, noté $a^{-1} \pmod b$

u et v sont calculables par l'algorithme d'Euclide étendu (cf. Moodle)

Indicatrice d'Euler : $\phi(n)$ = nombre d'entiers entre 1 et n premiers avec n

Si n est premier $\phi(n) = n-1$

Rappel : RSA

- Cryptographie à clé publique
  - RSA (**R**ivest, **S**hamir, **A**dleman, 1977)

Le Récepteur génère les clés:
Il prend deux *grands* nombres premiers *p* et *q* gardés secrets

clé "publique":

- $n = p*q$
- $\phi(n)=\phi(p)\phi(q)= (p-1)(q-1)$
- Soit e premier avec $\phi(n)$
- $d = e^{-1} \bmod \phi(n)$ par l'algorithme d'Euclide étendu

càd : $(e.d) \bmod \phi(n) = 1$

clé "privée" : e (élément inversible)

Rappel : RSA

Exemple: ***Récepteur*** génère les clés

Soit p = 9767 , q = 12347 , (p et q premiers), n = p*q = 120593149

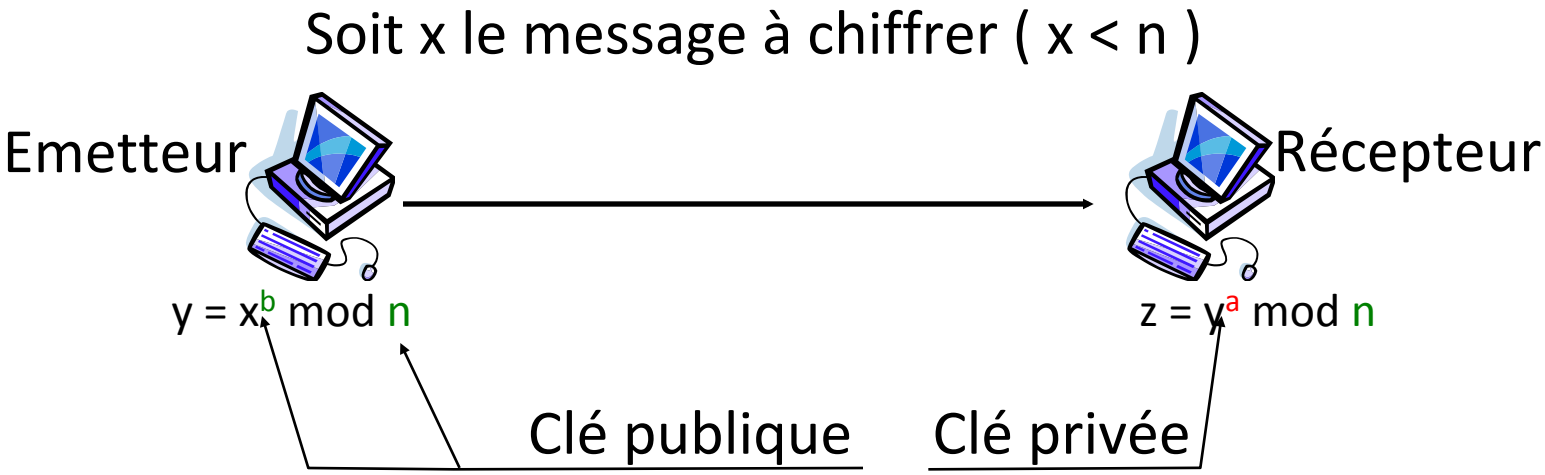
$\phi(n) = (p-1)(q-1) = 120571036$

e = 17345973 et $d = e^{-1} \bmod \phi(n) = 7013393$

On a bien $e \cdot d \bmod \phi(n) = 1$

Récepteur publie (d , n) = (7013393 , 120593149)

clé privée
clé publique



Exemple

clé privée
clé publique

Emetteur chiffre le message : « hello » et le découpe en blocs de 24 bits

h	e	l	l	o	
104	101	108	108	111	32

$X = (01101000\ 01100101\ 01101100)_2 = (6841708)_{10}$

$Y = 6841708^{7013393} \bmod 120593149 = 12409058$

Récepteur reçoit Y et calcule $Z = 1240905^{17345973} \bmod 120593149$

$Z = 6841708 = 104*256^2 + 101*256 + 108$

104	101	108
1101000	1100101	1101100
h	e	l

Devinette (un peu de maths)

- Soient des objets en nombre x inconnu.
- Si on les range par 3 il en reste 2
- Si on les range par 5, il en reste 3
- Si on les range par 7, il en reste 2

- Combien a-t-on d'objets ?

Les restes chinois

$n_1, n_2, \ldots, n_k$ premiers entre eux 2 à 2
Soit $n = n_1 . n_2 . \ldots . n_k$

Soient des objets en nombre x inconnu.
Si on les range par 3 il en reste 2
Si on les range par 5, il en reste 3
et si on les range par 7, il en reste 2

Soit le système :

$x = a_1 \pmod{n_1}$
$x = a_2 \pmod{n_2}$
……..
$x = a_j \pmod{n_k}$

$n_1, n_2, \ldots, n_j$, $a_1, a_2, \ldots, a_j$, connus

<u>Théorème</u> :
Il existe une solution unique (modulo n) avec

$x = \sum a_i * e_i$ avec

$e_i = k_i * (k_i^{-1} \pmod{n_i})$ et $k_i = n/n_i$

Démonstration

- n_i et $k_i = n/n_i$ sont premiers entre eux
- Donc il existe u_i et v_i tels que : $u_i\, n_i + v_i\, k_i = 1$ (Bezout) avec v_i = inverse de k_i modulo n_i
- En posant $e_i = v_i\, k_i$, on a :

  $e_i = 1$ modulo n_i

  $e_i = 0$ modulo n_j avec j différent de i
- Une solution particulière est :

  $x = \sum a_i * e_i$

Exemple

Soient des objets en nombre x inconnu.
Si on les range par 3 il en reste 2
Si on les range par 5, il en reste 3
et si on les range par 7, il en reste 2

$x = 2 \pmod 3 \Rightarrow k_1 = (3*5*7)/3 = 35$
$x = 3 \pmod 5 \Rightarrow k_2 = (3*5*7)/5 = 21$
$x = 2 \pmod 7 \Rightarrow k_3 = (3*5*7)/7 = 15$

$x = \sum a_i * e_i$ avec
$e_i = k_i * (k_i^{-1} \pmod{n_i})$ et $k_i = n/n_i$

on obtient

$n_1 = 3$ et $k_1 = 35 \Rightarrow k_1^{-1} \pmod 3 = 2$ puisque $2*35 = 1 \pmod 3$
$n_2 = 5$ et $k_2 = 21 \Rightarrow k_2^{-1} \pmod 5 = 1$ puisque $1*21 = 1 \pmod 5$
$n_3 = 7$ et $k_3 = 15 \Rightarrow k_3^{-1} \pmod 7 = 1$ puisque $1*15 = 1 \pmod 7$

une solution pour x est $x = 2*35*2 + 3*21*1 + 2*15*1 = 233 \pmod{105}$

et les solutions sont tous les entiers congrus à 233 modulo 105, c'est-à-dire à 23 modulo 105.

x = 23 ou 23+105 ou 23+105+105 ou …

RSA-ChineseRemainderTheorem

- $n = p * q$, p et q premiers (rôles des n_1 et n_2 du CRT)
- Soit à calculer $S = x^e \pmod n$

- CRT : But = accélérer le calcul avec des modulo plus petits

- On calcule

  $S_1 = x^e \pmod p$ (S1 joue le rôle du a_1 précédent, p celui du n1)

  $S_2 = x^e \pmod q$ (S2 ……., q celui du n2)

  $k_1 = n/n1 = q$, $k_2 = n/n2 = p$

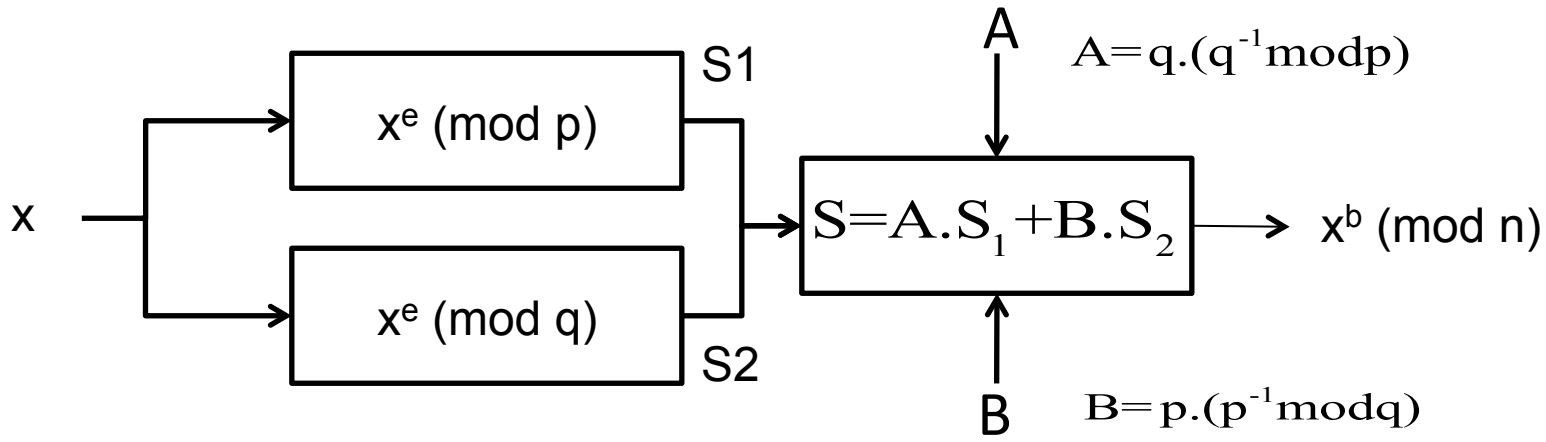
- En appliquant le résultat du théorème (restes chinois)

  $S = A . S1 + B . S2$ avec

  $A = q . (q^{-1} \bmod p)$ et $B = p. (p^{-1} \bmod q)$

  A et B sont précalculés

RSA-CRT architecture



Exemple

- n= 5 x 7 ; $\phi(n)$ = (5-1) (7-1) = 24
- e = 11 (premier avec 24)

- Soit x = 4 à chiffrer donc à calculer $y = 4^{11} \pmod{35}$

1/ Approche "classique"

A faire (sans calculette) calculer : $4^{11} \pmod{35}$

2/ Approche CRT

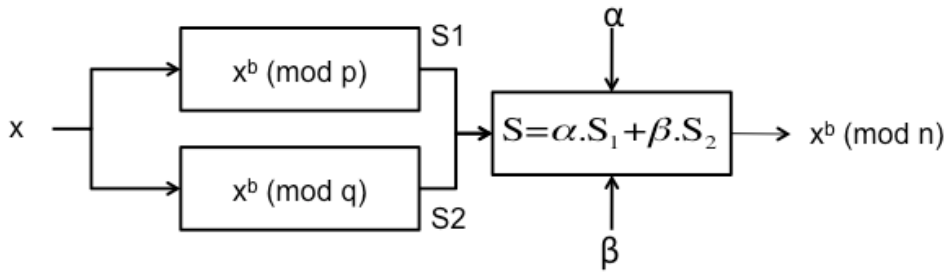
Inverse de 5 (mod7) = 3 : 3 x 5 = 15 = 1 (mod 7)

Inverse de 7 (mod5) = 3 : 3 x 7 = 21 = 1 (mod5)

Calculer $4^{11} \pmod{5}$ = S1

Calculer $4^{11} \pmod{7}$ = S2

Calculer : 7x3xS1 + 5x3xS2 (éventuellement réduire modulo 35)



Calculer $p^{-1} \pmod q$

- Rappels :

  Identité de Bezout :

  Soit p et q $\in$ N, $\exists$ u et v $\in$ Z / p.u + q.v = pgcd (p,q)

  Si p et q premiers entre eux (en part. si q premier) :

  $p.u + q.v = 1 \Leftrightarrow p.u = 1 - q.v \Leftrightarrow p.u = 1 \pmod q \Leftrightarrow$

  $u = p^{-1} \pmod q$

  Comment calculer l'inverse modulaire $p^{-1} \pmod q$?

Algorithme d'Euclide étendu

- Exemple : calculer l'inverse de 7 modulo 17

$17 = 2 \times 7 + 3$

$7 = 2 \times 3 + 1$ (pgcd)

En reprenant à l'envers :

$1 = 7 - 2 \times 3$

$= 7 - 2 \times (17 - 2 \times 7)$

$= 7 - 2 \times 17 + 4 \times 7$

$= 5 \times 7 - 2 \times 17$

$\Rightarrow 5 \times 7 = 1 \pmod{17} \Rightarrow 5 = 7^{-1} \pmod{17}$

Exercice

- Inverse de 30 mod 17 ?

Solution

- Déjà: 30 (mod 17) = 13 (mod 17)

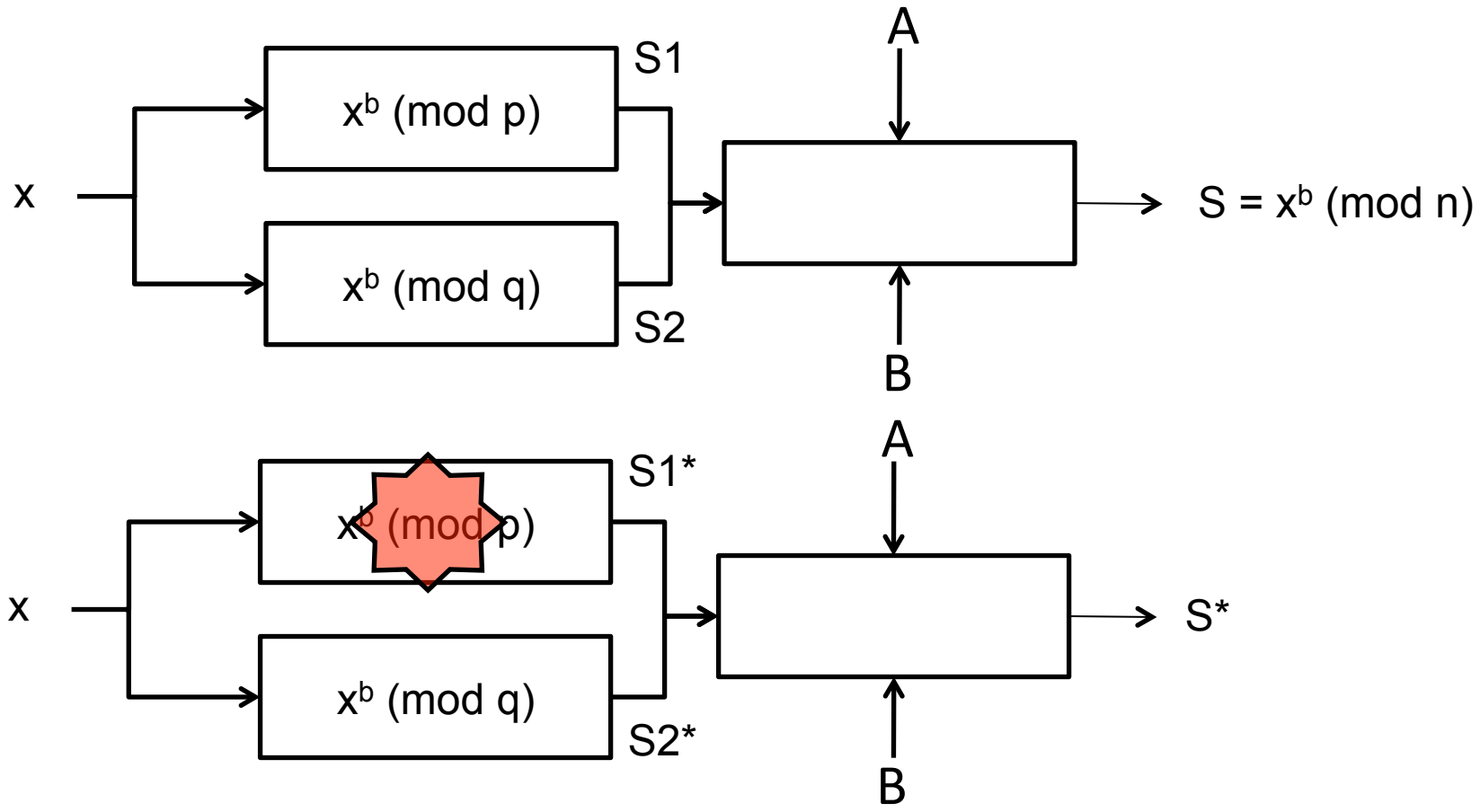
- 17 = 1x13 + 4
- 13 = 3x4 + 1

- 1 = 13 – 3x4 = 13 – 3x(17 – 1x13) = 4x13 - 3x17

- => 13^{-1} (mod 17) = 4

- Preuve : 4x13 = 52 = 51+1 = 3x17 + 1

Attaque : phase 1



On code 2 fois le même message, une fois sans faute, une fois avec faute.

S1 ≠ S1* S2 = S2*

Attaque : phase 2

Rappel : $S = q \cdot (q^{-1} \bmod p) \cdot S1 + p \cdot (p^{-1} \bmod q)$

$S \pmod q = [q \cdot (q^{-1} \bmod p) \cdot S1 + p \cdot (p^{-1} \bmod q)] \pmod q$
$= [q \cdot (q^{-1} \bmod p) \cdot S1] \pmod q + [p \cdot (p^{-1} \bmod q)] \pmod q$
$= 0 + [p \cdot (p^{-1} \bmod q)] \pmod q$

De même : $S^* \pmod q = [p \cdot (p^{-1} \bmod q)] \pmod q$

Donc $S \pmod q = S^* \pmod q$ càd q divise $(S-S^*)$

Par contre

$S \pmod p = [q \cdot (q^{-1} \bmod p) \cdot S1 + p \cdot (p^{-1} \bmod q)] \pmod p$
$= [q \cdot (q^{-1} \bmod p) \cdot S1] \pmod p$

$S^* \pmod p = [q \cdot (q^{-1} \bmod p) \cdot S1^* + p \cdot (p^{-1} \bmod q)] \pmod p$
$= [q \cdot (q^{-1} \bmod p) \cdot S1^*] \pmod p$

Comme $S1 \neq S1^*$, p ne divise pas $(S-S^*$

En résumé :

p et q premiers, n = p*q

q divise $(S-S^*)$ et p ne divise pas $(S-S^*)$

Donc PGCD $(S-S^*, n = pq) = q$

On a n, S et S*, on en déduit q, on en déduit p. On a tout

Exercice

- n = 35 = p*q (imaginez à la place, un très grand nombre n produit de 2 nombres premiers), e =11
- Trouver p et q (5 et 7 ici évidement)
- Soit x = 4 à coder càd calculer S = 4^{11} (modulo 35)
- Sans erreur :

  p=5, p^{-1} mod 7 = 3 => A = 3*5 = 15 =1 mod 7

  q=7, q^{-1} mod 5 = 3 => B = 7*3 = 21 =1 mod 5

  A = 3*7 =21

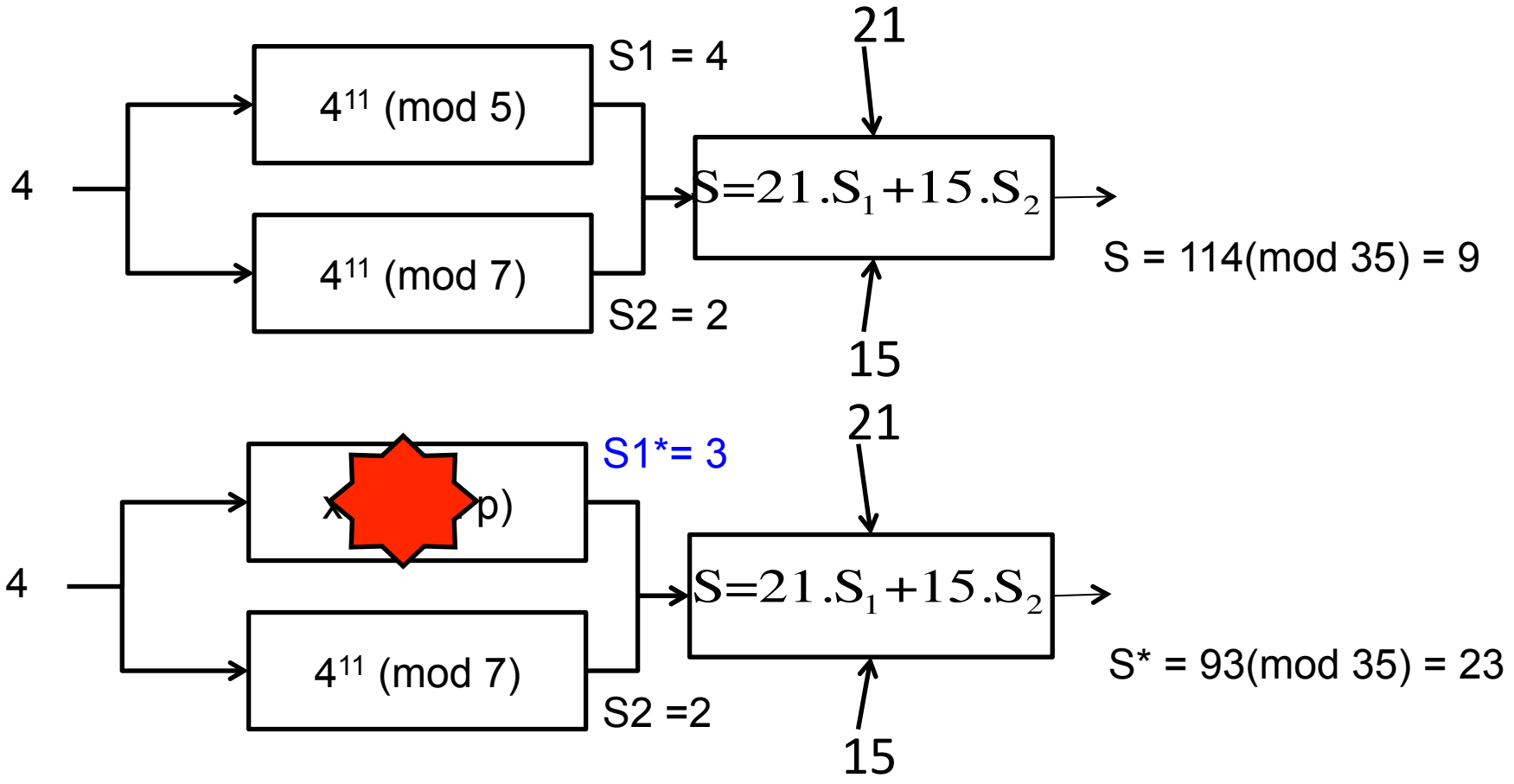
  B = 3*5 =15

S1 = 4^{11} (modulo 5) = 4

S2 = 4^{11} (modulo 7) = 2

S = 4 * 21 + 15 * 2 (modulo 35) = 114 (modulo 35) = 9

Exercice


4
4^{11} (mod 5)
4^{11} (mod 7)
S1 = 4
S2 = 2
21
15
$S=21.S_1+15.S_2$
S = 114(mod 35) = 9
4
p)
4^{11} (mod 7)
S1*= 3
S2 =2
21
15
$S=21.S_1+15.S_2$
S* = 93(mod 35) = 23

Récapitulatif

- Connus

  n = 35 =p*q

  S = 9

  S*= 23

- PGCD (23-9, 35) = PGCD (14, 35) = 7 = q
- n = 35 = p*7 => p = 5

Attaque : conditions

- Les conditions de l'attaque :
  - Pouvoir crypter 2 fois le même message (qui est quelconque)
  - Pouvoir injecter une erreur dans un seul des deux calcul d'exponentiation
- Pas d'hypothèse sur le type d'erreur !

Les attaques sur l'AES

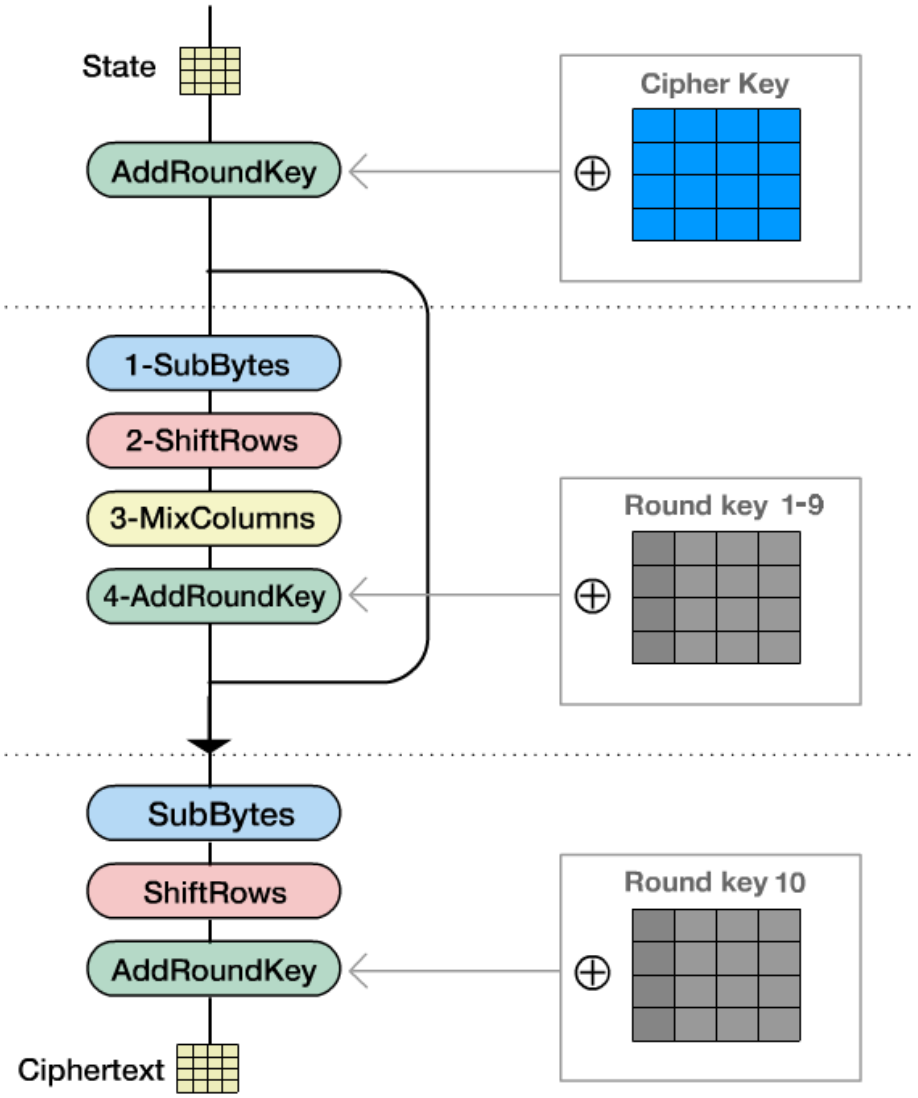
- Giraud (Oberthur Card System) 2003
  - Attaque sur un bit sur entrée dernier SubBytes
  - Attaque sur KeySchedule (K9, K8) puis datapath (M8)

- Chen et Yen 2003
  - Attaque sur KeySchedule (K9, 2 fois K8)
  - Contre-mesures présentées

- Attaques sur datapath (MixColumns)
  - Dusart, Vivolo et Letourneux 2003
  - Piret et Quisquater 2003

- Safe error (Blömmer et Seifert) 2003
  - Attaque bit
  - 2 attaque octet
  - Implémentation xtimes

AES


State
AddRoundKey
Cipher Key
1-SubBytes
2-ShiftRows
3-MixColumns
4-AddRoundKey
Round key 1-9
SubBytes
ShiftRows
AddRoundKey
Round key 10
Ciphertext

Les attaques sur l'AES

	Target	Focalization	#Faults
[BS03]	Data(M_0)	Bit	128
[CNSM03]	Data 7*(M_8)+Key 4*(K_9)	Byte	11
[Gir03]	Data 16*(M_9)	Bit	16
[PjQ03]	Key 4*(K_9)+4*(K_{10})+Data 4*(M_8)	Byte	12
[PjQ03]	Data 4*(M_9) or 1*(M_8)	Byte	4 or 1
[CT05]	Round counter	Round counter value	1
[PMC+11]	Round counter	Byte	1
[DLV03]	Data 4*(M_9)	Byte	4
[RM07]	Data before first subbyte	Bit or Byte	16
[KQ08]	K_7	Byte	1
[AMT]	Data at 8^{th} round input	Byte	1

Giraud (Oberthur Card System) 2003 : Attaque sur un bit sur entrée dernier SubBytes

L'attaque bit de Giraud

- Faute :
  - On attaque 1 bit sur un octet avant le dernier SubBytes
  - L'octet est faux à la sortie
  - On cherche à trouver un octet de la clef

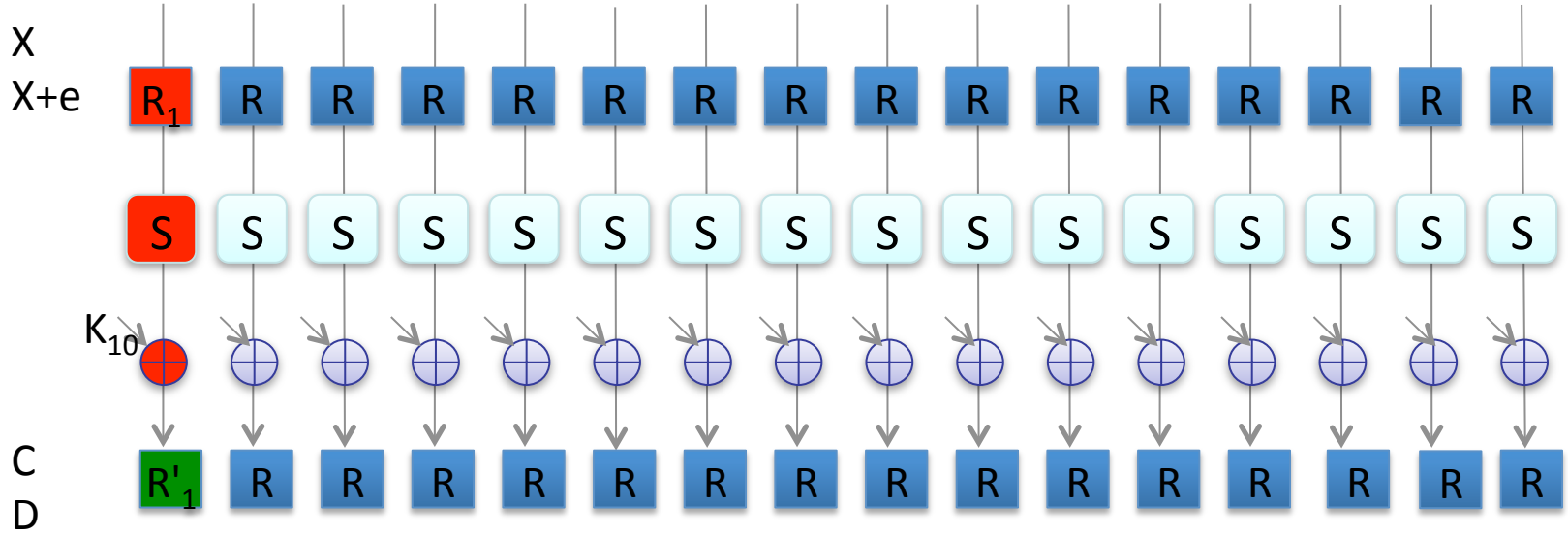
- Attaque :
  - On réalise deux exécutions, une normale et une fautée, avec le même texte clair non connu
  - On "xore" les deux résultats, un seul octet diffère
  - La position de l'octet nous donne l'octet sur lequel l'attaque a eu lieu (InvShiftRows)
  - On fait des hypothèses sur le bit faux e et X, on calcule les candidats

$$C = SB(X) \oplus K10$$

$$D = SB(X \oplus e) \oplus K10$$

$$\Delta = SB(X) \oplus SB(X \oplus e)$$

Attaque 'bit' de Giraud
Dernière ronde de l'AES



$$C = SB(X) \oplus K10$$

$$D = SB(X \oplus e) \oplus K10$$

$$\Delta = SB(X) \oplus SB(X \oplus e)$$

L'attaque bit de Giraud

- On essaie (en simulation) les 256 valeurs de X et les 8 valeurs de e / $SB(X) \oplus SB(X \oplus e) = C \oplus D = \Delta$
- 8 solutions en moyenne pour X
- $K_{10} = C \oplus SB(X)$ (8 solutions)
- On refait avec un autre X
  - On trouve l'octet de clef avec la proba 82%
  - Avec un troisième X, P = 99%
- Fautes requises : moins de 50 pour la clé complète

Attaque : conditions

- Les conditions de l'attaque :
  - Pouvoir crypter 2 fois le même message (qui est quelconque)
  - Pouvoir injecter une erreur
    - sur un seul bit
    - au début de la dernière ronde

Exercice

- Le premier octet du crypté d'un message est 0x71 (hexa).
- Le premier octet du crypté du même message avec injection d'une faute lors de la dernière ronde est 0xaa.
- Les autres octets sont identiques à ceux du premier crypté

- Le premier octet du crypté d'un autre message est 0xab
- Avec une faute, le premier octet du crypté est 0x39

- Quelle est le premier octet de la clef ? voir moodle

- https://www.tutorialspoint.com/compile_c_online.php

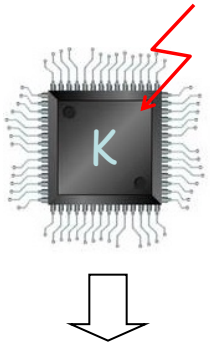
Mise en œuvre des attaques en fautes

- ✓ Par perturbation de la tension d'alimentation
- ✓ Par perturbation de l'horloge
- ✓ Par perturbation de la température
- ✓Attaques Électromagnétiques
- ✓Injection laser
- ✓ Injection de particules

Techniques d’injection de fautes

- Techniques d’injection de fautes non invasives.
  - Théorie (violations de setup et de délai)
  - Réalisations expérimentales
- Techniques d’injection de fautes semi-invasives.
  - L’injection laser (effet photoélectrique)
  - Injection mono octet

Partie expérimentale



Technique d'injection

Chiffrés fautés, side channels, comportement, etc.

Extraction informations

Méthodes

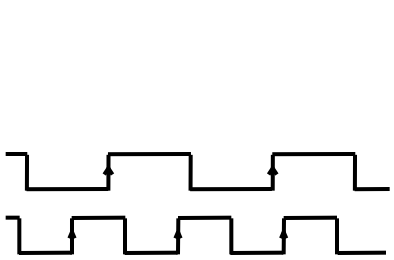
- Réduction nb. rondes
- Analyse différentielle de fautes
- Safe error

Modèle de faute

- Instant d'injection
- Bit /Octet
- Aléatoire / Donnée choisie

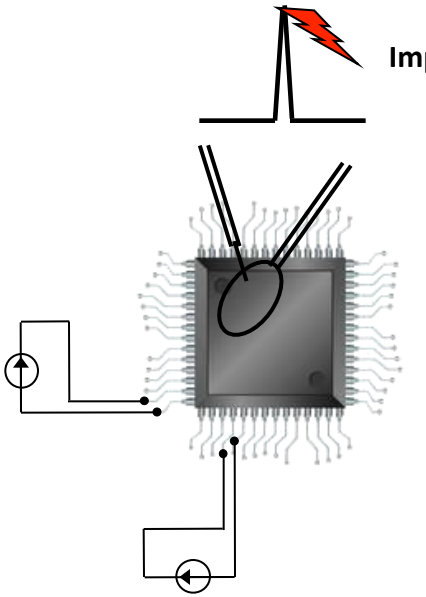
La technique d'injection employée doit permettre l'injection de fautes respectant le modèle de faute requis.

- Attaques non invasives : ne requérant pas l'ouverture du boîtier.

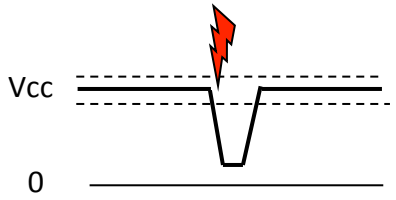


Over clocking

Clk "glitch"



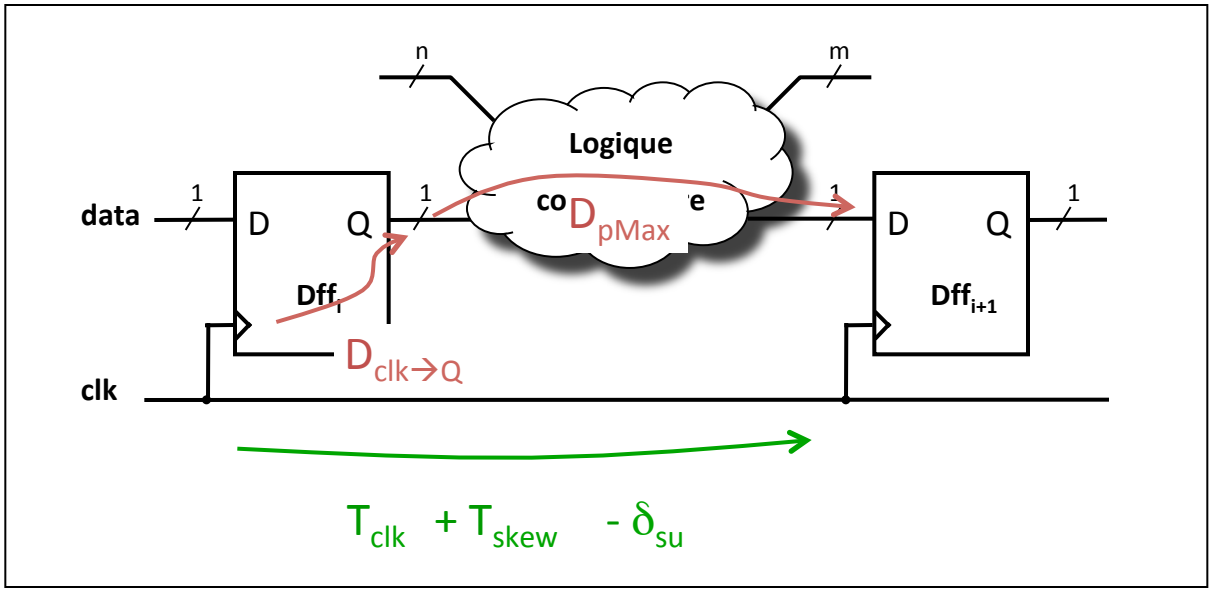
Augmentation de T°



Modification de V_{dd}

"glitch" d'alimentation

- Contrainte temporelle des circuits synchrones.



data arrival time = $D_{clk\rightarrow Q} + D_{pMax}$

data required time = $T_{clk} + T_{skew} - \delta_{su}$

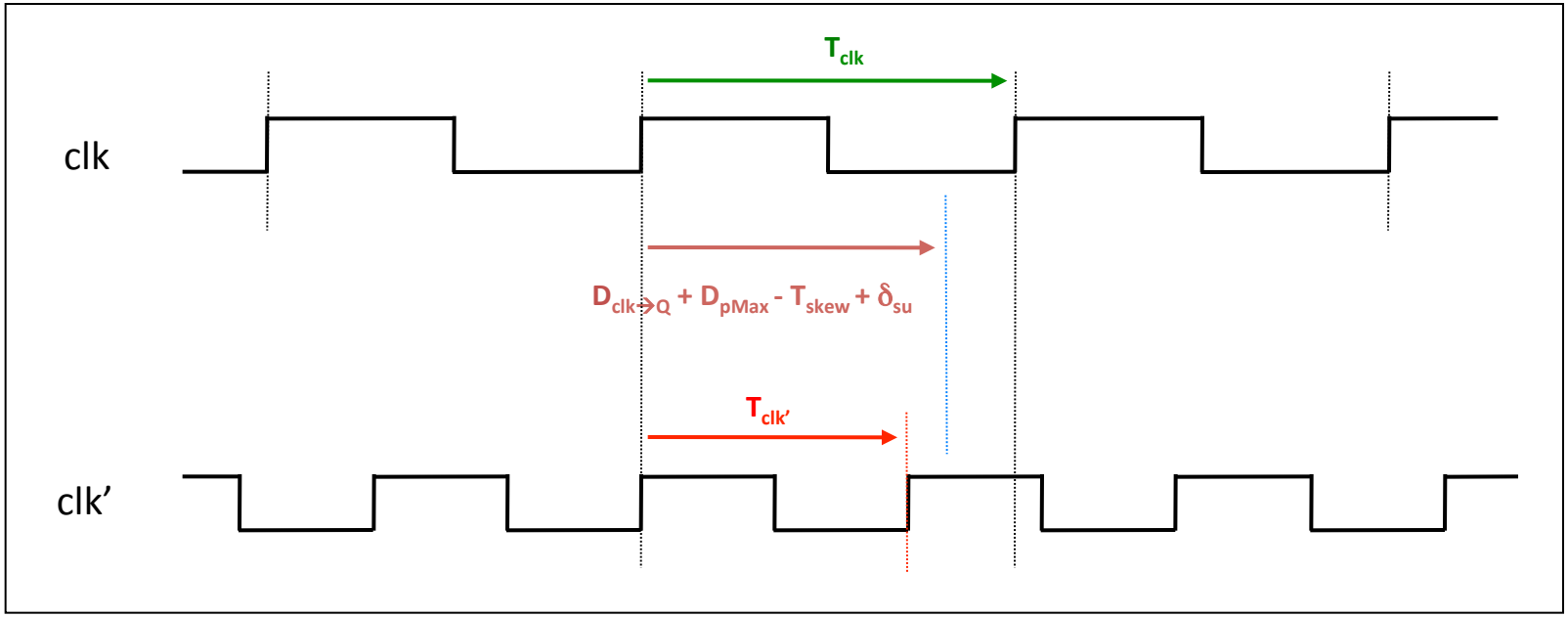
$\Rightarrow$ $T_{clk} > D_{clk\rightarrow Q} + D_{pMax} - T_{skew} + \delta_{su}$

Injection de faute par violation de temps de setup / délai

- Overclocking.

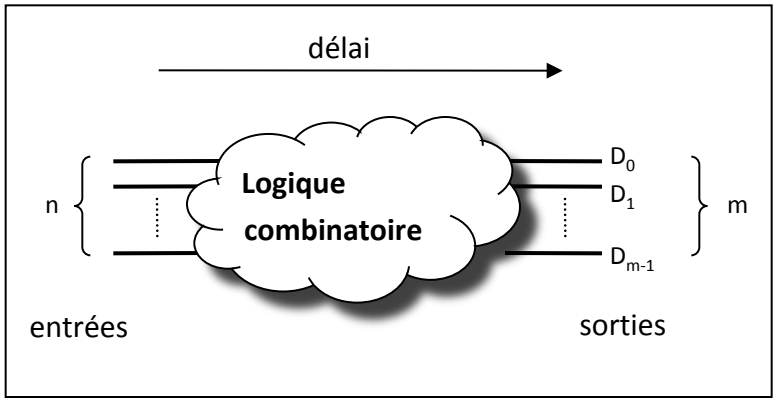
Approche classique : décroissance progressive de T_{clk} jusqu'à obtenir une violation de temps de setup.

$$T_{clk'} < D_{clk \rightarrow Q} + D_{pMax} - T_{skew} + \delta_{su} < T_{clk}$$



Injection de faute par violation de temps de setup / délai

- Temps de propagation – Chemin critique



sorties = f (entrées)

f fonction logique

chaque D_i possède son propre temps de propagation

Localisation des fautes :

délai > $T_{clk'}$ – setup time – $D_{clk \rightarrow Q}$ + T_{skew}

Chemin critique = max des tps de propagation

Le temps de propagation dépend :

- **des niveaux logiques (0 / 1)**
  → le temps de propagation change avec les entrées
- de la tension d'alimentation
- de la température

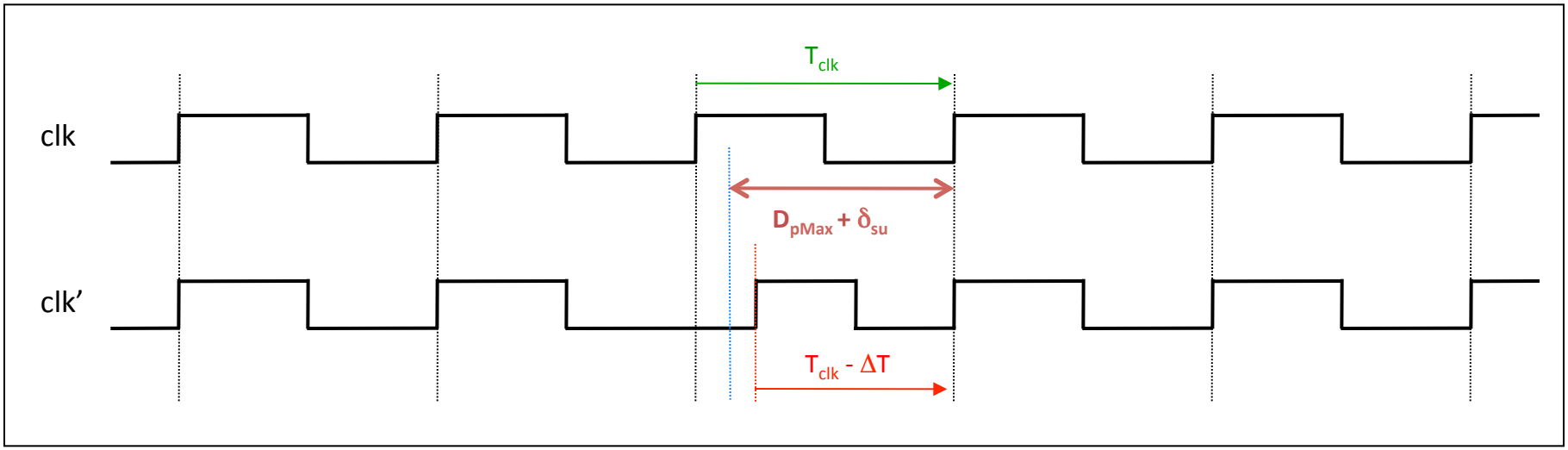
} **permet de modifier l'endroit d'injection**

Injection de faute par violation de temps de setup / délai

- Overclocking (suite).

⇨ limitation : injection de fautes potentiellement à chaque cycle d'horloge.

- *Glitch* d'horloge – Modification locale d'une période.



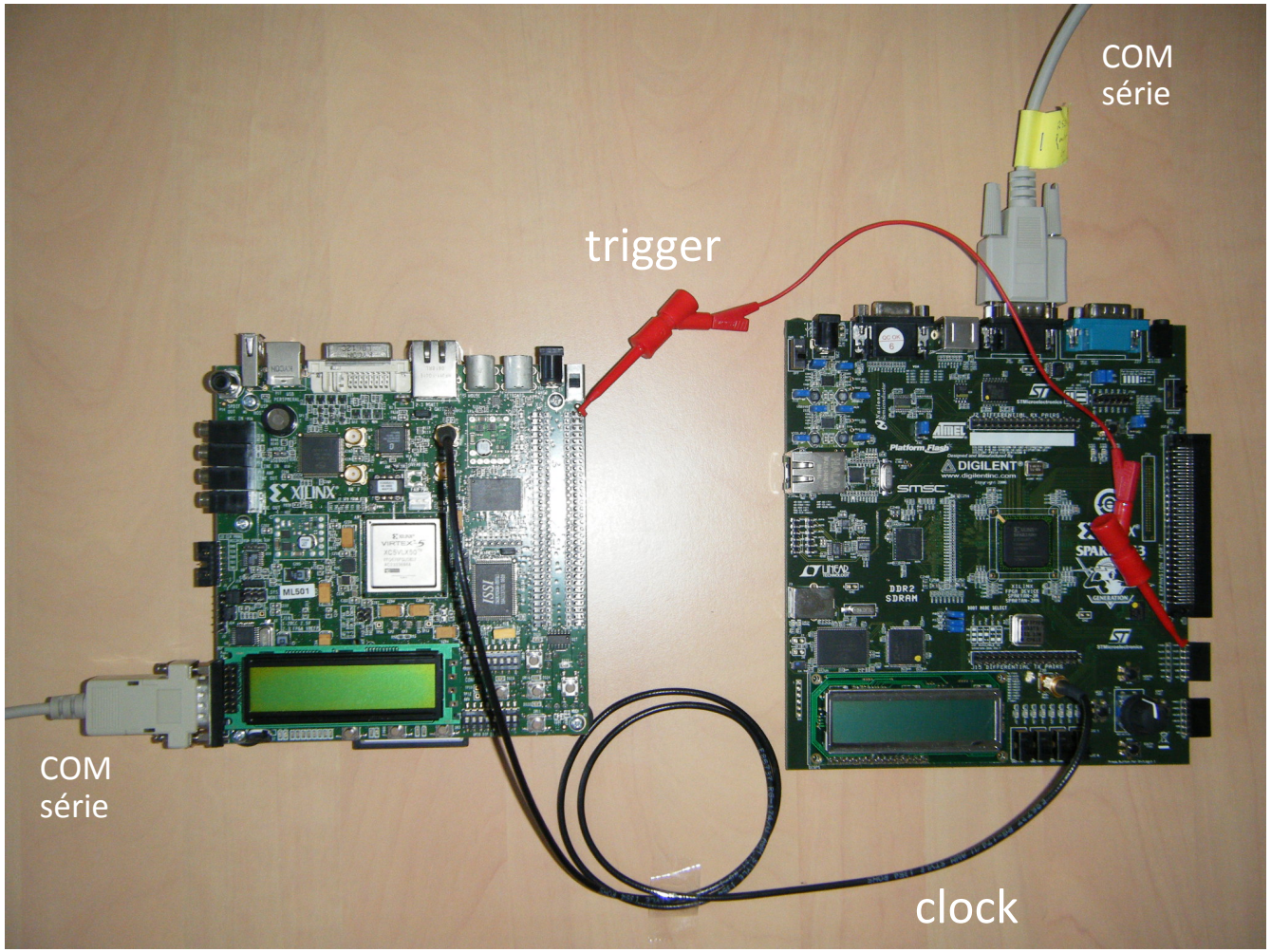
⇨ Choix du cycle d'injection.

⇨ Contrôle fin de la nature des fautes injectées ($\Delta T = 35$ ps).

Injection de faute par violation de temps de setup / délai

- Dispositif expérimental

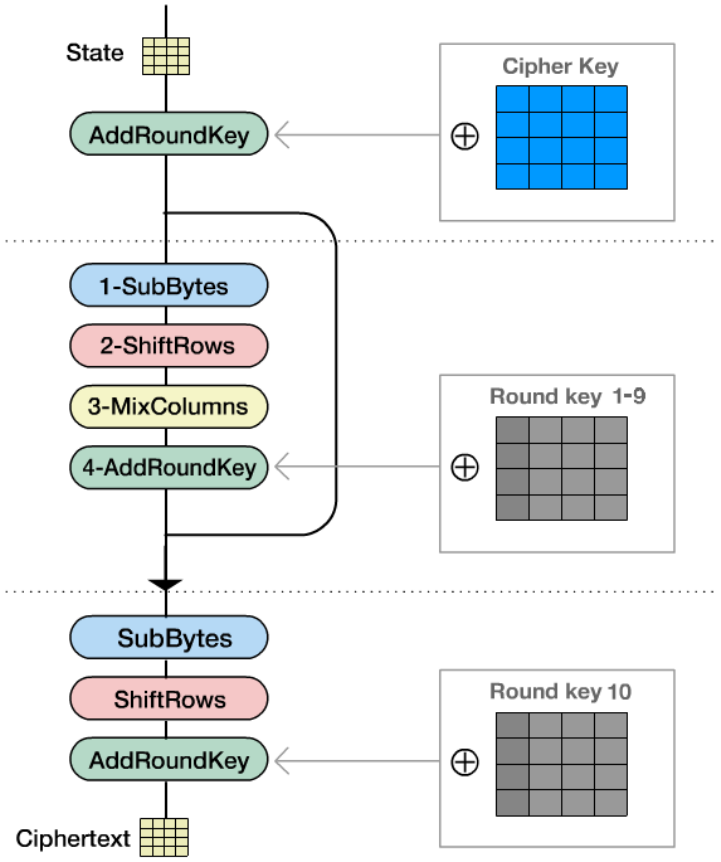
Clock generation board

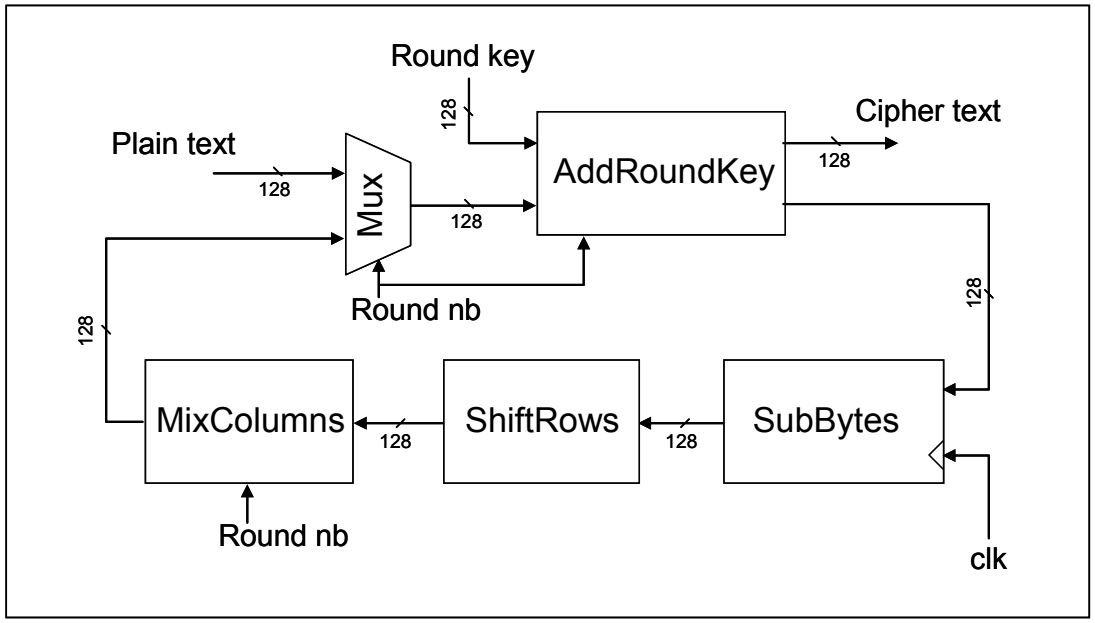


AES board

Injection de faute par violation de temps de setup / délai

- Cible : AES matériel.

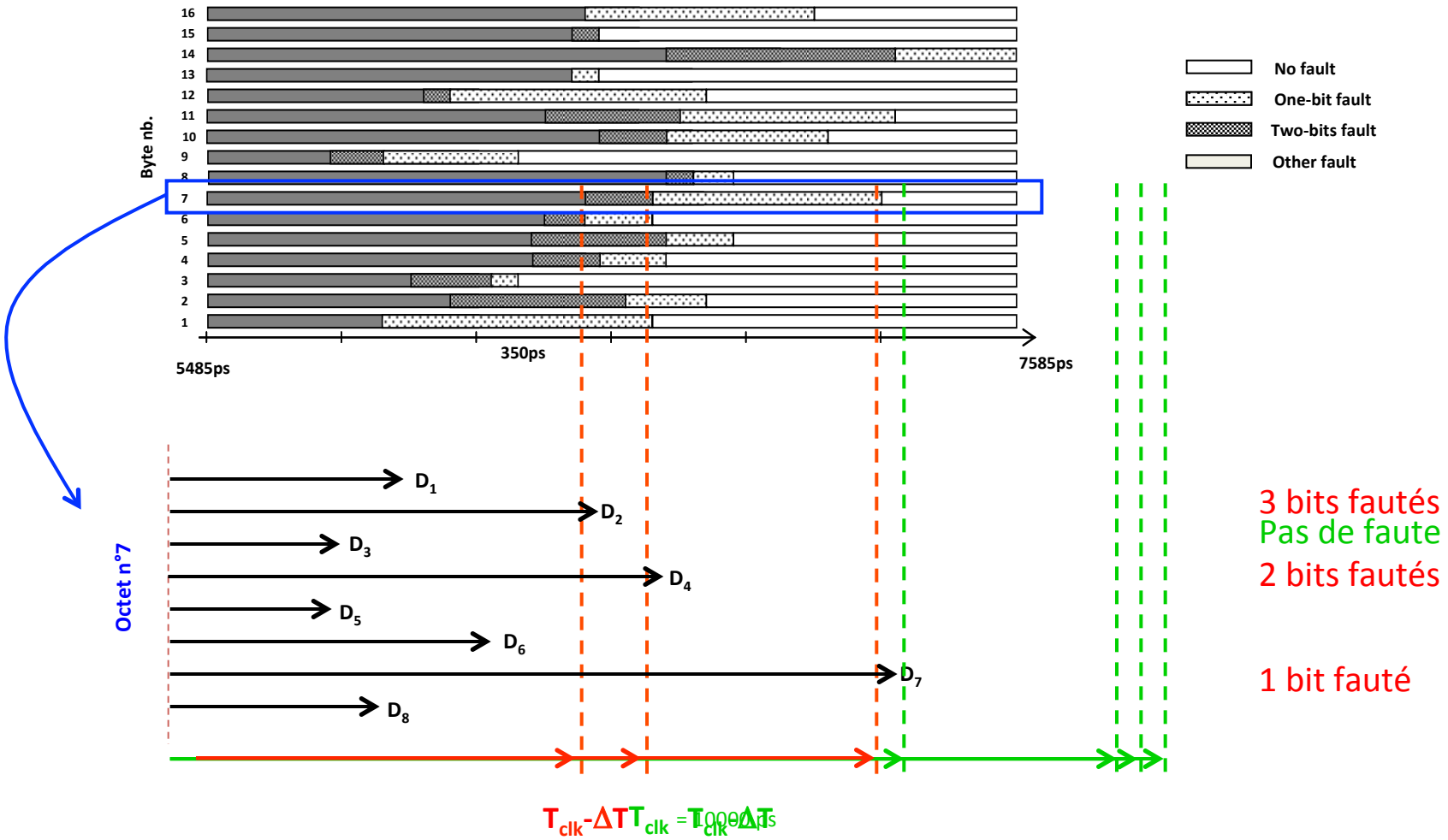




- Registre en entrée du SubBytes
- Architecture en boucle (i.e. chemin critique)

Injection en ronde finale ($f_{clk, nom}$ = 100 MHz)

Diminution progressive de T_{clk} (ΔT = 35 ps)



- Glitch d'horloge.

Accès CLK requis

Contrôle de la focalisation : excellent

Faute mono-bit	> 90%
Fautes 1 puis 2 bits	> 70%
Fautes 1,2 puis 3 bits	> 50%

Contrôle de l'instant d'injection : total (choix du cycle)

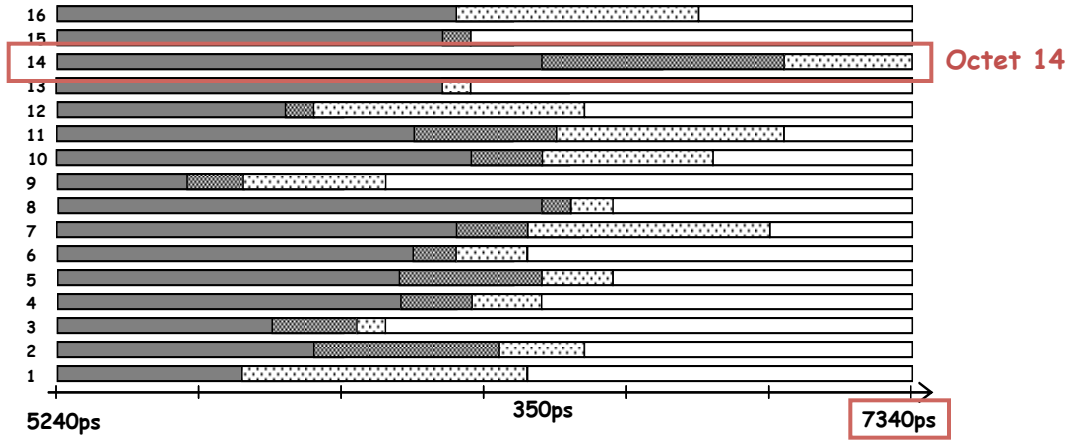
Type de fautes : aléatoire

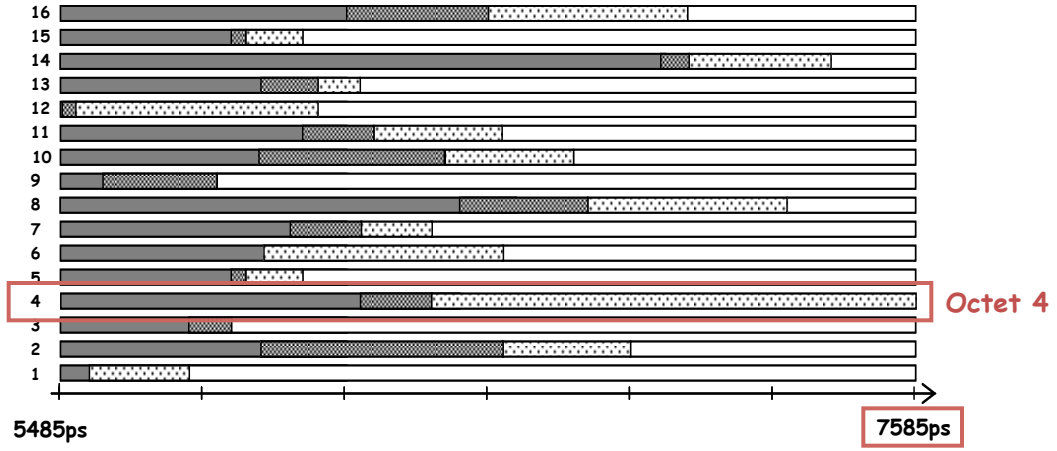
Facilité d'emploi : aisée (plateforme numérique)

Coût : faible (<1000 €)

- Glitch d'horloge.

Contrôle de la localisation : par variation du texte clair





Même clef

Texte clair différent

- Glitch d'horloge.

Résultats obtenus pour 12000 essais :

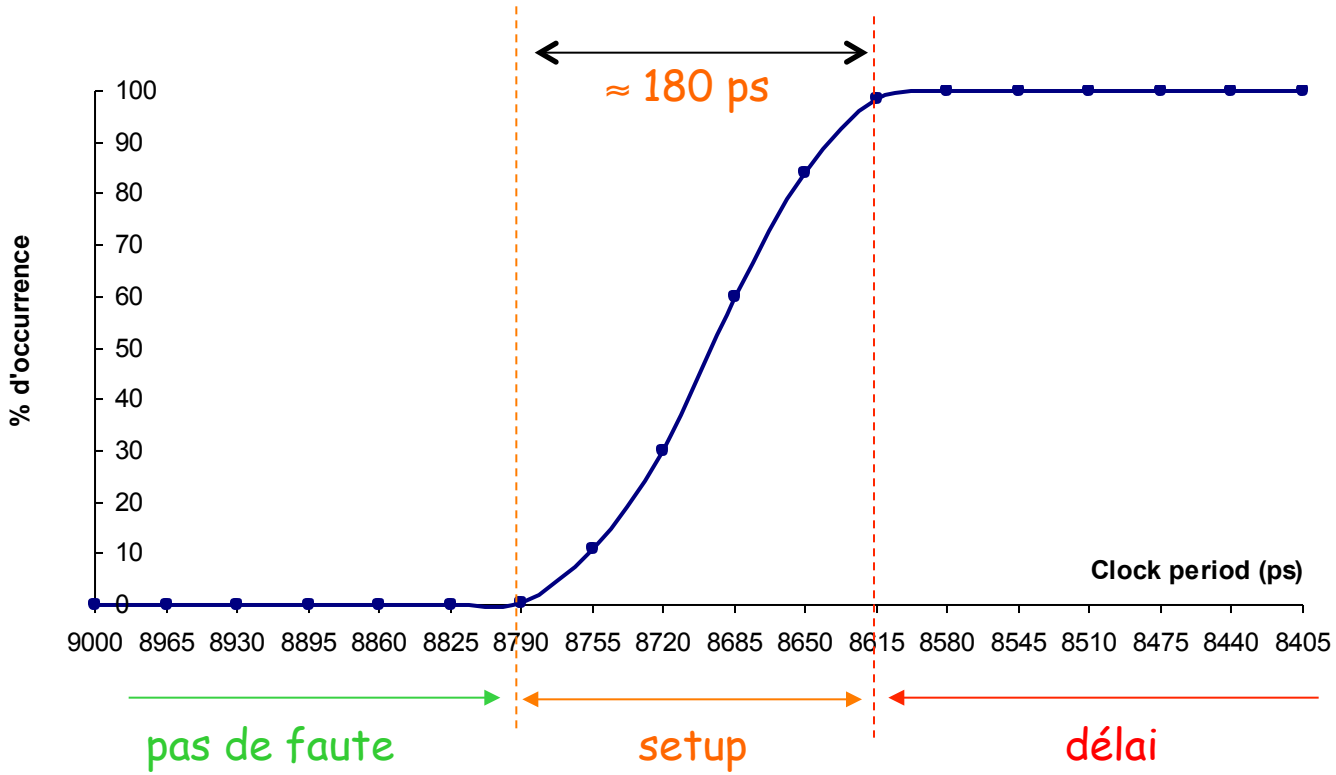
	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10	Byte 11	Byte 12	Byte 13	Byte 14	Byte 15
bit0	0	0	77	893	0	0	56	1	6	0	1402	0	438	746	22	0
bit1	0	9	1554	11	17	0	1	176	0	0	13	0	0	1	0	7
bit2	0	216	0	0	0	0	0	107	1	11	2	2	0	10	10	21
bit3	0	0	0	629	2	0	1	0	0	1	0	56	0	0	0	663
bit4	0	32	0	275	33	0	0	3	0	0	0	0	222	147	0	29
bit5	0	0	312	33	23	0	0	1290	22	0	0	2	368	9	0	406
bit6	225	690	0	69	83	5	0	0	0	486	1	0	0	0	5	3
bit7	0	10	0	3	95	0	0	62	12	43	0	0	0	0	0	0
Total	225	957	1943	1913	253	5	58	1639	41	541	1418	60	1028	913	37	1129

< 10
< 100
≥ 100

Contrôle de la localisation : possible (dépendant de l'implémentation)

- Glitch d'horloge.

Reproductibilité : bonne (mais non absolue)



Injection de faute par violation de temps de setup / délai

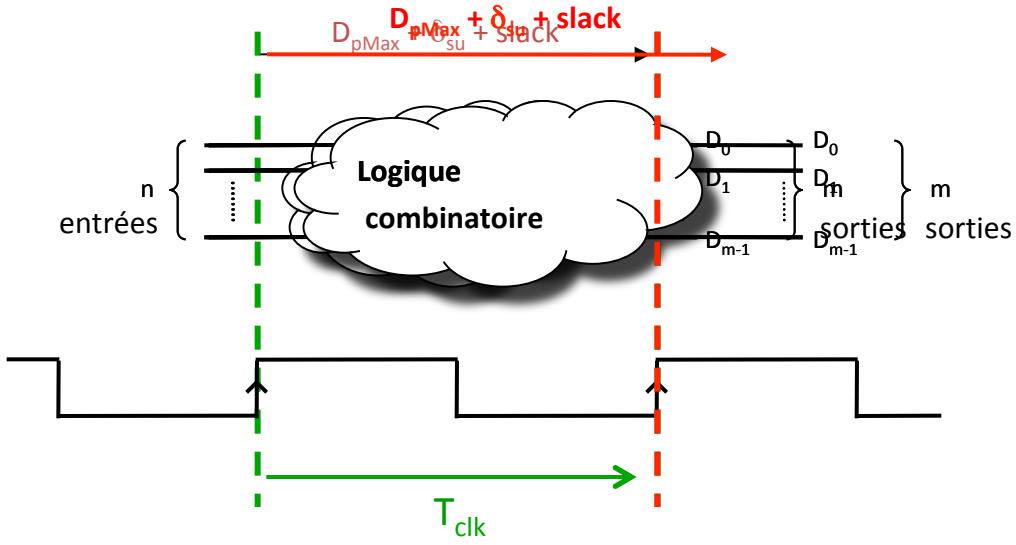
- Injection de faute par diminution de la tension d'alimentation.
  (à fréquence nominale)

Tension d'alimentation ↘

⇨ D_{pMax} ↗ ($D_{clk \rightarrow Q}$, δ_{su}, $|T_{skew}|$) ↗

$$T_{clk} < D_{clk \rightarrow Q} + D_{pMax} - T_{skew} + \delta_{su}$$

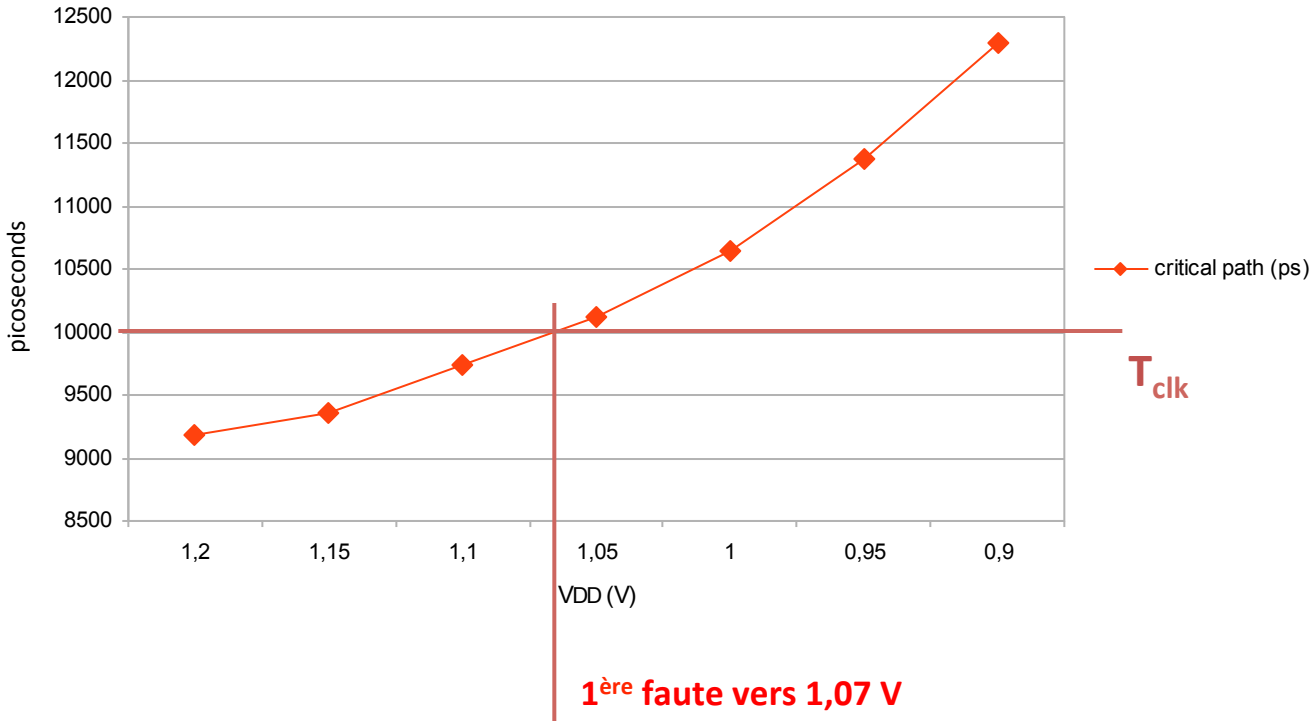
Injection de fautes à chaque cycle d'horloge



Injection de faute par violation de temps de setup / délai

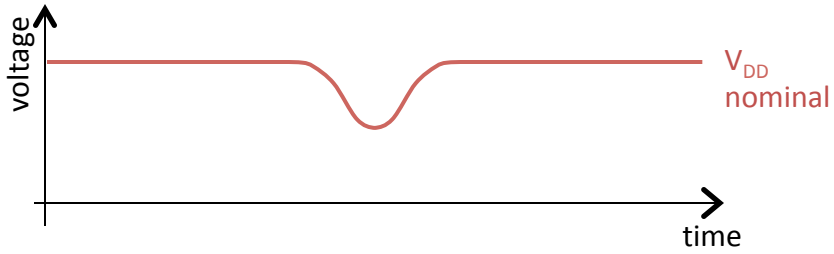
- Injection de faute par diminution de la tension d'alimentation.

Evolution du temps critique avec la tension d'alimentation :



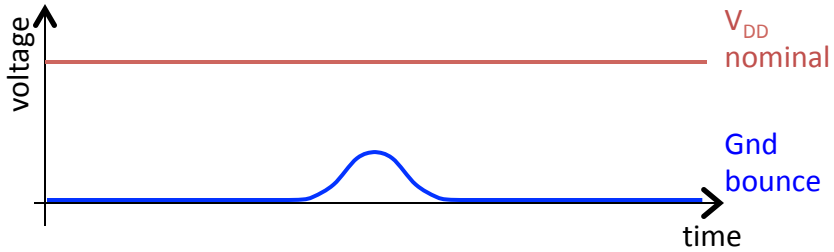
Injection de faute par violation de temps de setup / délai

- Glitch d'alimentation (à fréquence nominale)



Injection de faute durant le glitch

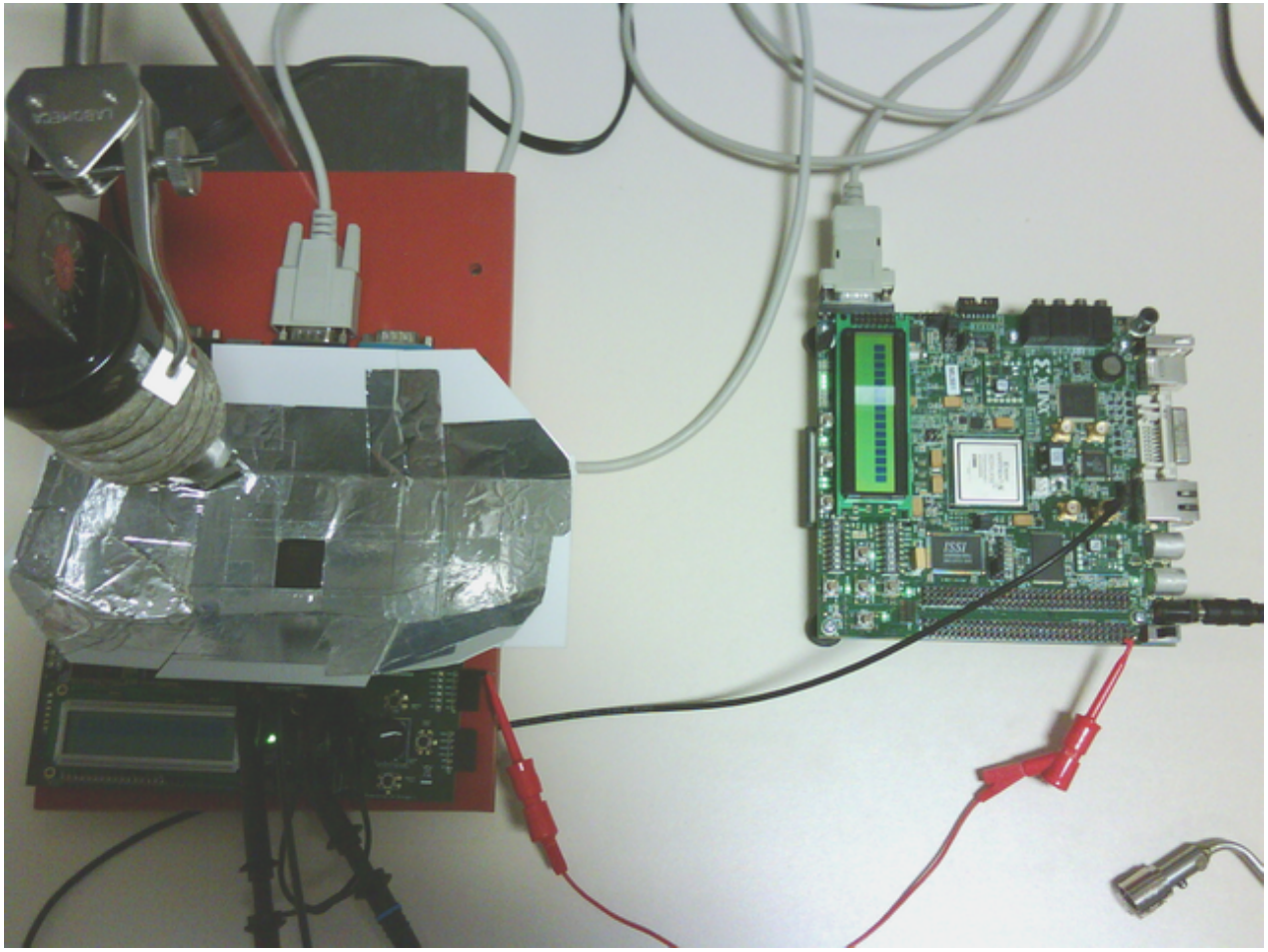
⇨ choix du cycle d'injection



Même mécanisme pour un glitch sur la masse (ground bounce)

Injection de faute par violation de temps de setup / délai

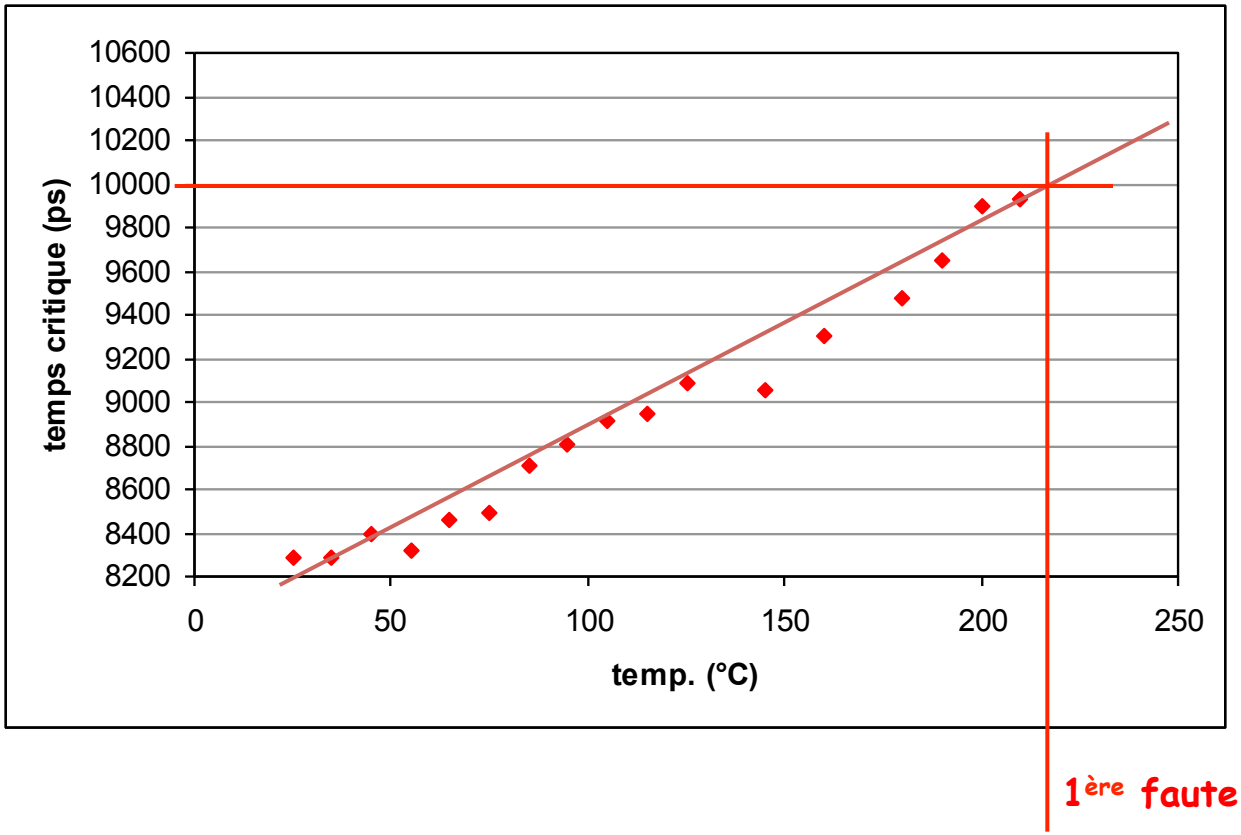
- Augmentation de la température (à fréquence nominale)



Injection de faute par violation de temps de setup / délai

- Augmentation de la température (à fréquence nominale)

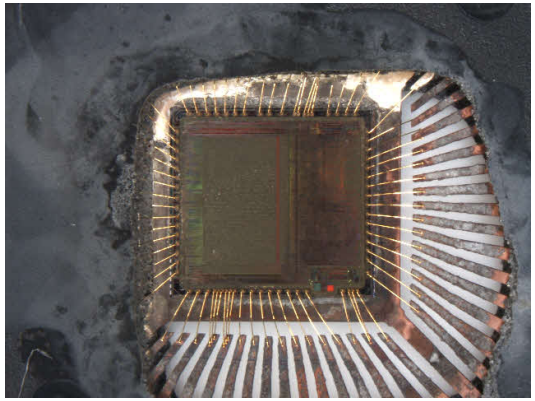
$\Rightarrow \quad D_{pMax} \nearrow \quad (D_{clk \rightarrow Q}, \delta_{su}, |T_{skew}| \quad) \nearrow$

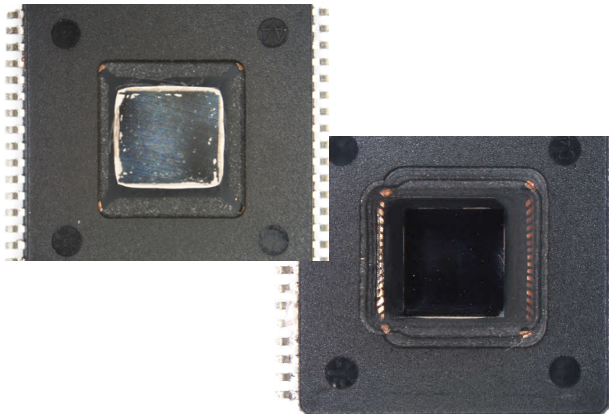


- Injection de faute par violation de temps de setup / délai :
  - Overclocking.
  - Glitch d'horloge.
  - Injection de faute par diminution de la tension d'alimentation.
  - Glitch d'alimentation.
  - Augmentation de la température.

⇨ Mécanisme d'injection similaire.

- Attaques semi-invasives : ouverture chimique/mécanique du boîtier.

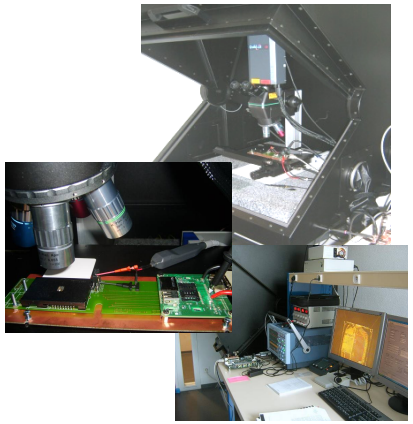




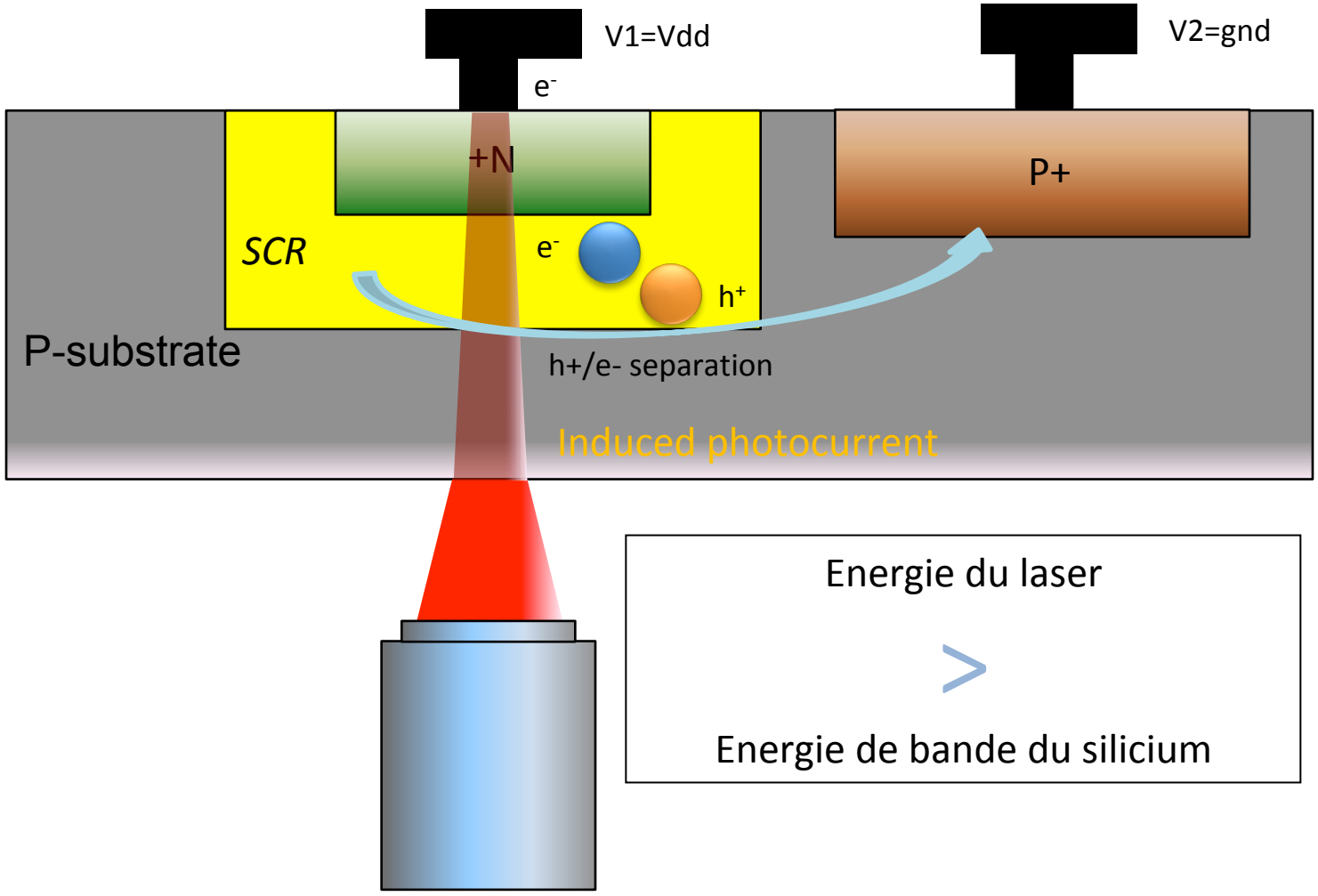
Techniques d'injection optique [Sko02] :

- Flash,
- Laser.

[Sko02]



Effet du laser sur le silicium - Effet photoélectrique

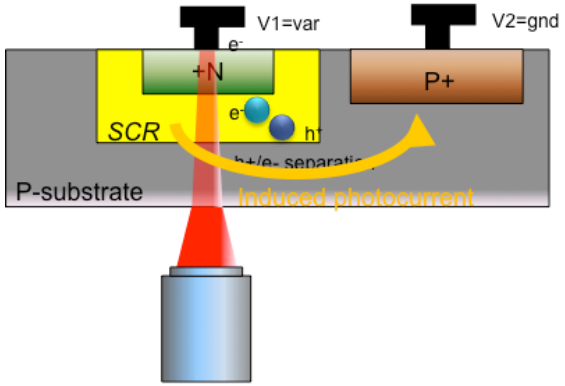


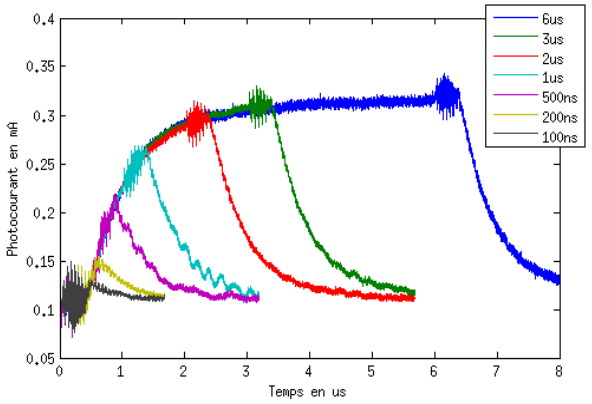
Energie du laser

>

Energie de bande du silicium

EFFET PHOTOELECTRIQE





$Iph = f(Power_{laser}, Area_{pn}, Distance_{pn}, Time_{laser}, ...)$

Transitoire de courant

=> Transitoire de tension

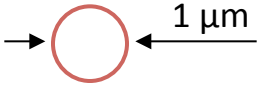
=> Faute/Erreur

- Effet du transitoire de tension :
  - propagation dans la logique sans mémorisation,
  - propagation dans la logique avec mémorisation,
  - inversion de l'état d'un point mémoire (registre, SRAM).

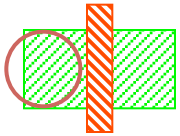
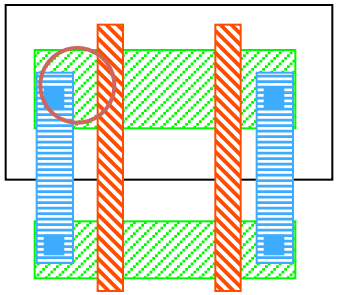
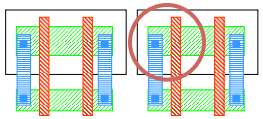
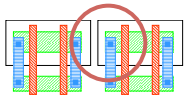
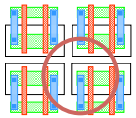
  ⇨ injection de faute

- Paramètres de réglage d'un laser :
  - longueur d'onde,
  - énergie,
  - taille du faisceau, (état de l'art 1 µm min.)
  - durée de l'impulsion,
  - gigue.

Contrôle de la focalisation :

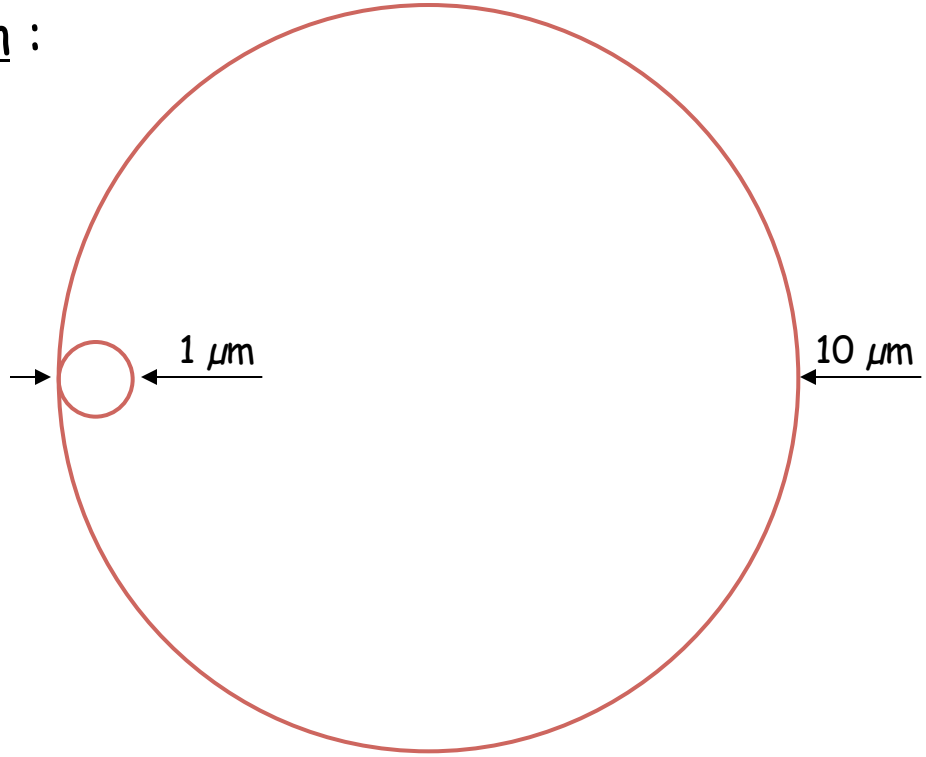


Spot laser

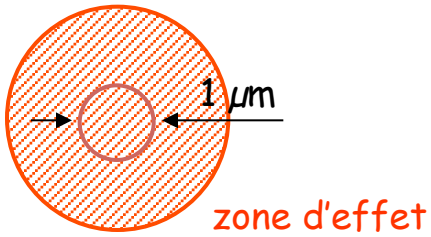
Technologie	Transistor MOS	SRAM
0.35 µm		
130 nm		
90 nm		
65 nm		

Contrôle de la focalisation :

- Taille du spot



- Zone d'effet
  Energie déposée



Contrôle de la localisation : lié au contrôle de la focalisation
platine (x,y,z)

Contrôle de l'instant d'injection : selon électronique de commande
et technologie (gigue).

Coût : élevé à très élevé (qqs 10^{aine} k€ à qqs 100^{aines} k€)

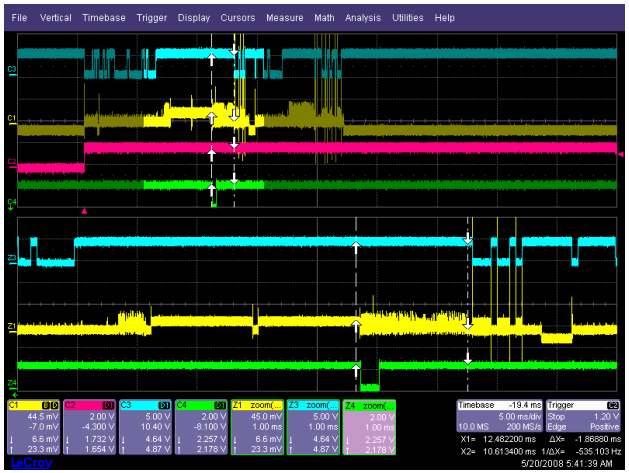
- Instrument de caractérisation sécuritaire (pour les plus chers)
  Multinationales, gouvernements, etc.

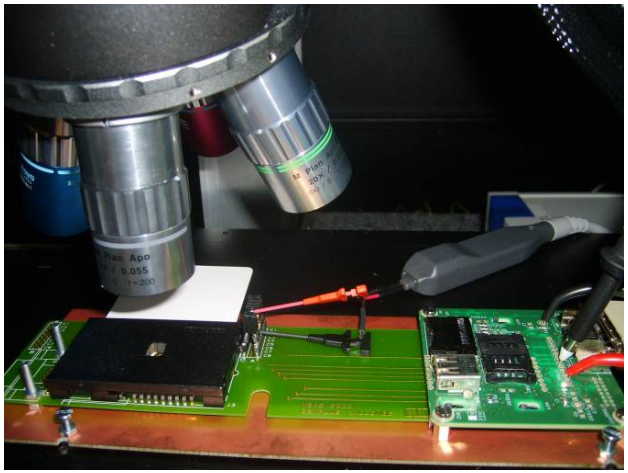
  Intérêt : reproductibilité, contrôle de la localisation/focalisation.

- Outil d'attaque pratique (pour les moins chers)
  Performances limitées (optiques inadaptées, pointeur laser, expertise, etc.).

Banc d’injection laser

- Laser (IR, Green, UV)
- XY stage
- Camera
- Oscilloscope
- Synchronization board
- ISO Reader (optional)



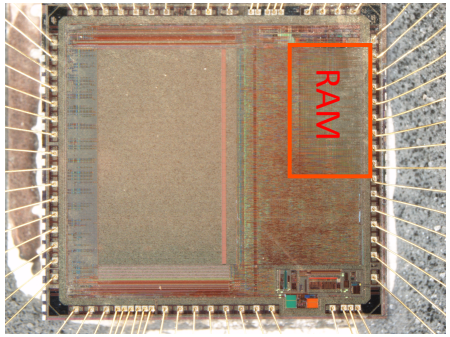


- Amélioration de la focalisation (injection mono-octet).

  Cadre : banc d'injection rudimentaire ($\varnothing_{spot}$ > 20 µm).

  ⇨ Gain en résolution spatiale dû au contrôle de l'instant d'injection

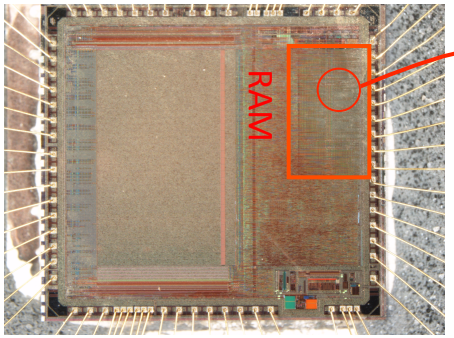
  Cible : implémentation AES sur microcontrôleur

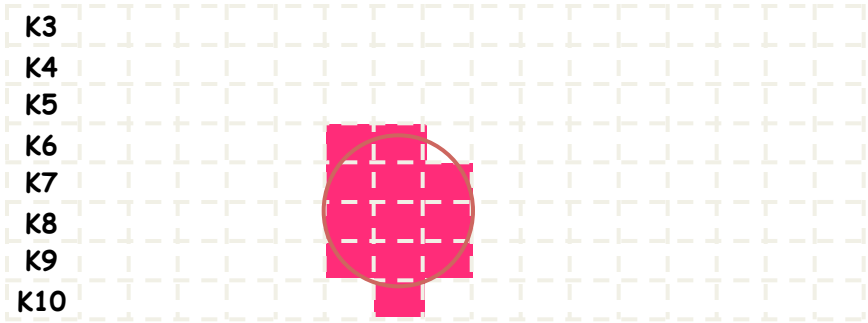


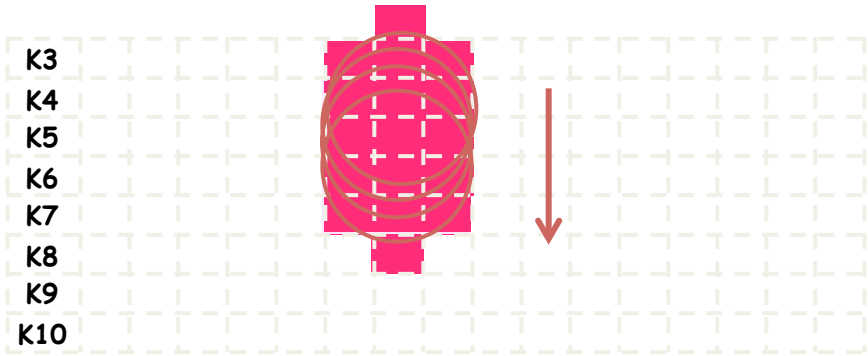
Attaque Piret – Quisquater :

fauter un octet avant le MixColumn de la ronde 9

⇨ fauter un octet de K8







Pas de déplacement : 0,1 µm

Injection en ronde 8
avant le AddRoundKey

⇨ Algorithmique : faute mono-octet

Physique : fautes multi octets

	Contrôle de			reproductibilité	coût	facilité d'emploi
	instant d'injection	localisation	focalisation			
Glitch d'horloge (numérique)	maximum	moyen	très bon	bonne	faible	très bonne
Glitch d'alimentation (analogique)	bon[1]	moyen	très bon	bonne	moyen	bonne
Overclocking Baisse V_{DD} Température	faible	moyen	bon	bonne	faible	bonne
Laser	bon[2]	très bon	très bon	bonne	élevé	bonne

[1] selon électronique de commande

[2] selon électronique de commande et technologie