

Challenges FCSC 2025

Cryptographie et Side Channel

St4ck0verfl0w

Redacted

June 10, 2025

France Cyber Security Challenge (FCSC)



- CTF organisé par l'**ANSSI** (individuel)
- Recrute l'équipe de France pour les championnats Européens

France Cyber Security Challenge (FCSC)



- CTF organisé par l'**ANSSI** (individuel)
- Recrute l'équipe de France pour les championnats Européens
- **10 jours d'épreuves** (18 au 27 mai), 99 challenges
- 10 catégories dont 6 classantes (**crypto, reverse, pwn, hardware, web, speerdun**, SCA, forensic, misc, intro)

France Cyber Security Challenge (FCSC)



- CTF organisé par l'**ANSSI** (individuel)
- Recrute l'équipe de France pour les championnats Européens
- **10 jours d'épreuves** (18 au 27 mai), 99 challenges
- 10 catégories dont 6 classantes (**crypto, reverse, pwn, hardware, web, speerdun**, SCA, forensic, misc, intro)
- 2046 inscrits (1848 ayant fait au moins une épreuve)
- <https://fcsc.fr> puis <https://hackropole.fr>

France Cyber Security Challenge (FCSC)



- CTF organisé par l'**ANSSI** (individuel)
- Recrute l'équipe de France pour les championnats Européens
- **10 jours d'épreuves** (18 au 27 mai), 99 challenges
- 10 catégories dont 6 classantes (**crypto, reverse, pwn, hardware, web, speerdun**, SCA, forensic, misc, intro)
- 2046 inscrits (1848 ayant fait au moins une épreuve)
- <https://fcsc.fr> puis <https://hackropole.fr>
- **Particularité cette année : le speedrun**

France Cyber Security Challenge (FCSC)



- CTF organisé par l'**ANSSI** (industriel)
- Recrute l'équipe de France pour les championnats Européens
- 10 jours d'épreuves (18 au 27 mai), 99 challenges
- 11 catégories dont 6 classantes (**crypto, reverse, pwn, hardware, web, speedrun**, SCA, forensic, misc, intro)
- 2046 inscrits (1848 ayant fait au moins une épreuve)
- <https://fcsc.fr> puis <https://hackropole.fr>
- Particularité cette année : le speedrun

FCSC 2025 Utilisateurs Classement Challenges Notifications Compte Paramètres

Challenges

FCSC 2025 has ended

Sauf mention contraire, le format des flags est FCSC{xxxx} où xxxx est une chaîne contenant des caractères ASCII imprimables.
Nous vous recommandons fortement de parcourir la FAQ.

feedback

Feedback	1 point	1m
153		

speedrun terminé

RSA-WTF	164 points	★
202		
Binary Lullaby	295 points	★
113		
Mikado	422 points	★★
49		
Polygraph	429 points	★
36		
Même Pamal	446 points	★★
35		
Back to BASIC	444 points	★
25		
Old is better than new	469 points	★
25		
Surinosaour (Dewaste)	467 points	★★
17		
eraise	479 points	★★
10		
La revanche de Sauron	487 points	★★
9		
Surinausor (Winds of the past)	493 points	★★★
5		
Happy Face	498 points	★★
4		

FCSC 2025 Utilisateurs Classement Challenges Compte Paramètres

Challenge 10 résolutions

AES Distrace

477 points

crypto

Vous avez la possibilité d'exécuter ce binaire qui effectue un chiffrement AES-128-ECB sur une entrée aléatoire, tout en observant la valeur d'un registre à une adresse fixe tout au long du calcul.

Note : le chemin `/app/` dans le fichier Python fourni (`aes-distrace.py`) est à adapter à votre environnement pour faire tourner l'épreuve localement.

nc chall.fcsc.fr 2151

[aes-distrace](#) [aes-distrace.c](#) [aes-distrace.py](#)
[libexeclog.so](#) [qemu-x86_64](#)

Flag Soumettre

Catch me if you can
25 points [intro](#)

yabof
25 points [intro](#)

Poète
25 points [intro](#)

Kzber
439 points [★★★](#)

La quête de l'anneau
100 points [★](#) [366](#) [★★](#) [156](#) [★★★](#) [25](#)

Fun with Hash
454 points [★★](#)

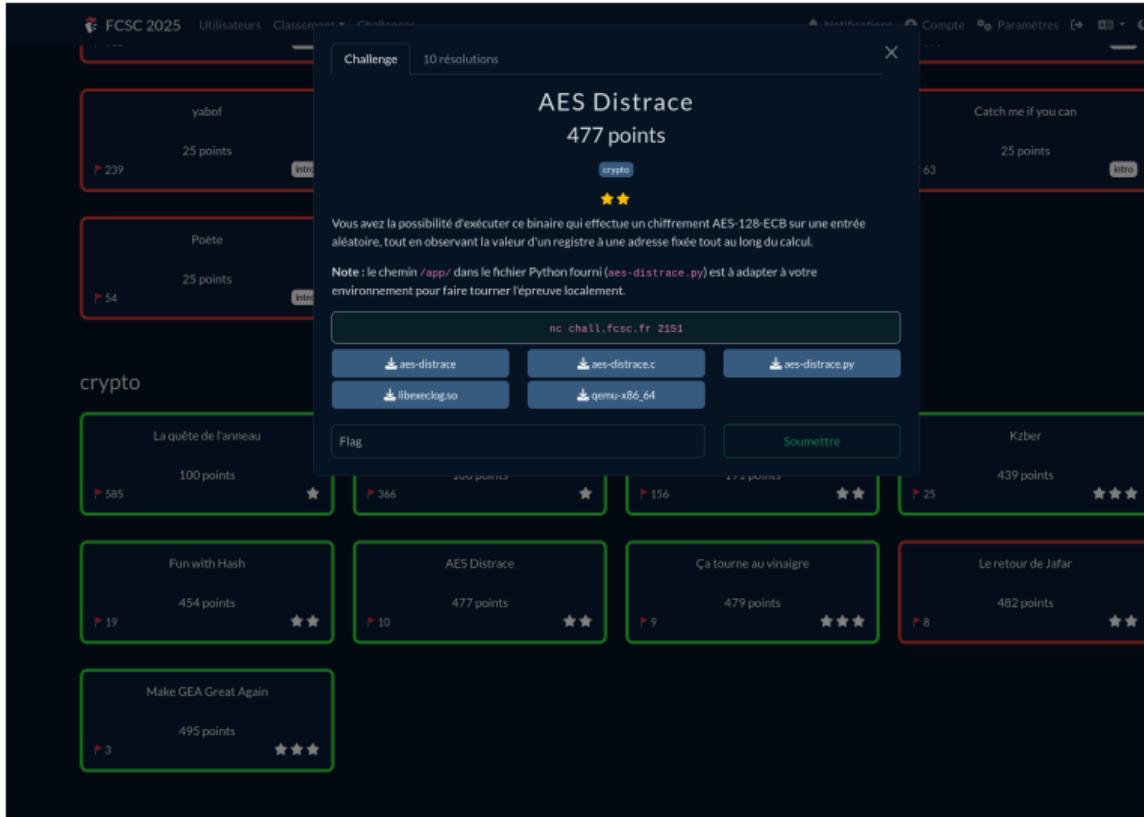
AES Distrace
477 points [★★](#)

Ça tourne au vinaigre
479 points [★★★](#)

Le retour de Jafar
482 points [★★](#)

Make GEA Great Again
495 points [★★★](#)

63 [intro](#)





FCSC 2025 Utilisateurs Classement Challenges Notifications Compte Paramètres

Top 3 par catégories

Catégories qualifiantes : crypto, reverse, web, pwn, hardware, speedrun.

Speedrun/Junior	Speedrun/Senior	Speedrun/Hors Catégorie
1 PoustouFlan	1 Dvorack	1 KLPP
2 meler	2 N04H	2 nikost
3 conflict	3 meouthon	3 base64(md5)FC...

Crypto/Junior	Crypto/Senior	Crypto/Hors Catégorie
1 PoustouFlan	1 SticksOverFlow	1 Red
2 Sevenas	2 bluesheet	2 base64(md5)FC...
3 ap6sh	3 GOM_Gear	3 goodguy...

Pwn/Junior	Pwn/Senior	Pwn/Hors Catégorie
1 meler	1 meouthon	1 Sh
2 PoustouFlan	2 Itarrow	2 nikost
3 D4h4Happy	3 skusk	3 FullHouse

Reverse/Junior	Reverse/Senior	Reverse/Hors Catégorie
1 TIR0K0SS	1 spikerot	1 008ILX
2 Prince2Lu	2 or1o	2 Edwardo
3 PoustouFlan	3 loukous24	3 A-Z but rev

Web/Junior	Web/Senior	Web/Hors Catégorie
1 Dawn	1 Warby	1 icefont
2 Vexited	2 Vozec	2 fwadzidor
3 Rooting	3 Minilucker	3 selsort

Hardware/Junior	Hardware/Senior	Hardware/Hors Catégorie
1 cartonne222	1 loukous24	1 base64(md5)FC...
2 PoustouFlan	2 Dvorack	2 MK37
3 Prince2 Lu	3 shd33	3 nikost

Sommaire

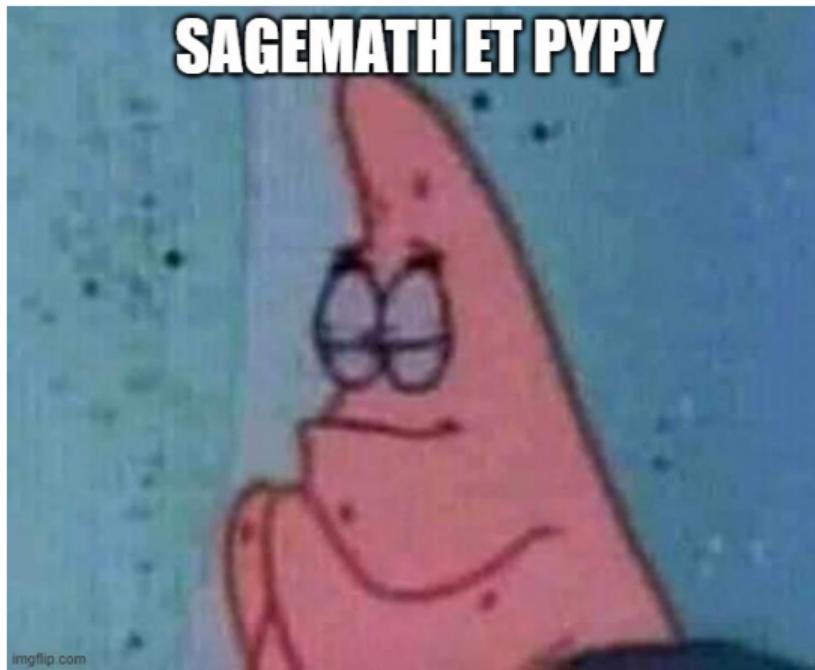
1. Side-channel

- 1.1 Differential Power Analysis
- 1.2 Attaques par fautes sur RSA et courbes elliptiques
- 1.3 Fuites de l'état interne AES

2. Cryptographie

- 2.1 Introduction : les classiques de ctf
- 2.2 Sur les fonctions de hashage md5 et sha1
- 2.3 Kyber et les anneaux quotients
- 2.4 Analyse différentielle et boomerang
- 2.5 Le schéma multivarié UOV
- 2.6 GEA1 et construction de backdoor

Les vrais héros



Differential Power Analysis

Introduction au side channel



crypto

- Objectif
 - Récupérer le PIN d'un portefeuille crypto



crypto



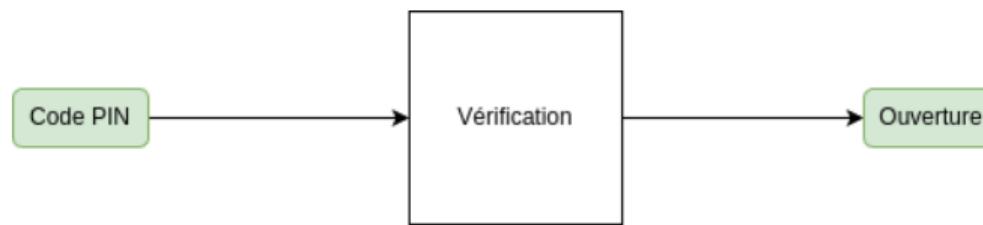
crypto

- Objectif
 - Récupérer le PIN d'un portefeuille crypto
- Données
 - Une trace courant pour chaque PIN

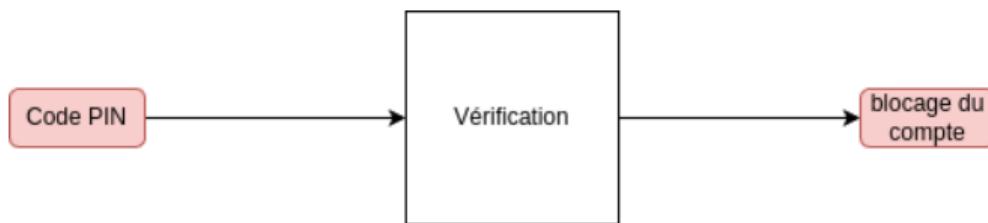


crypto

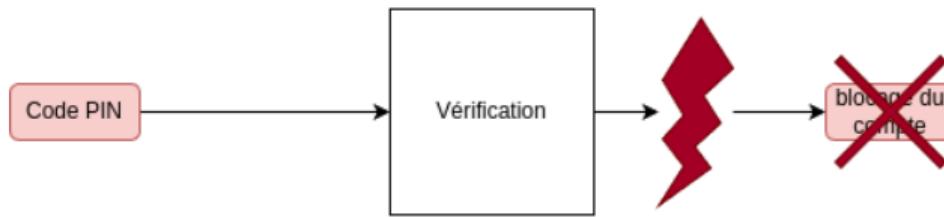
Faillock et bypass



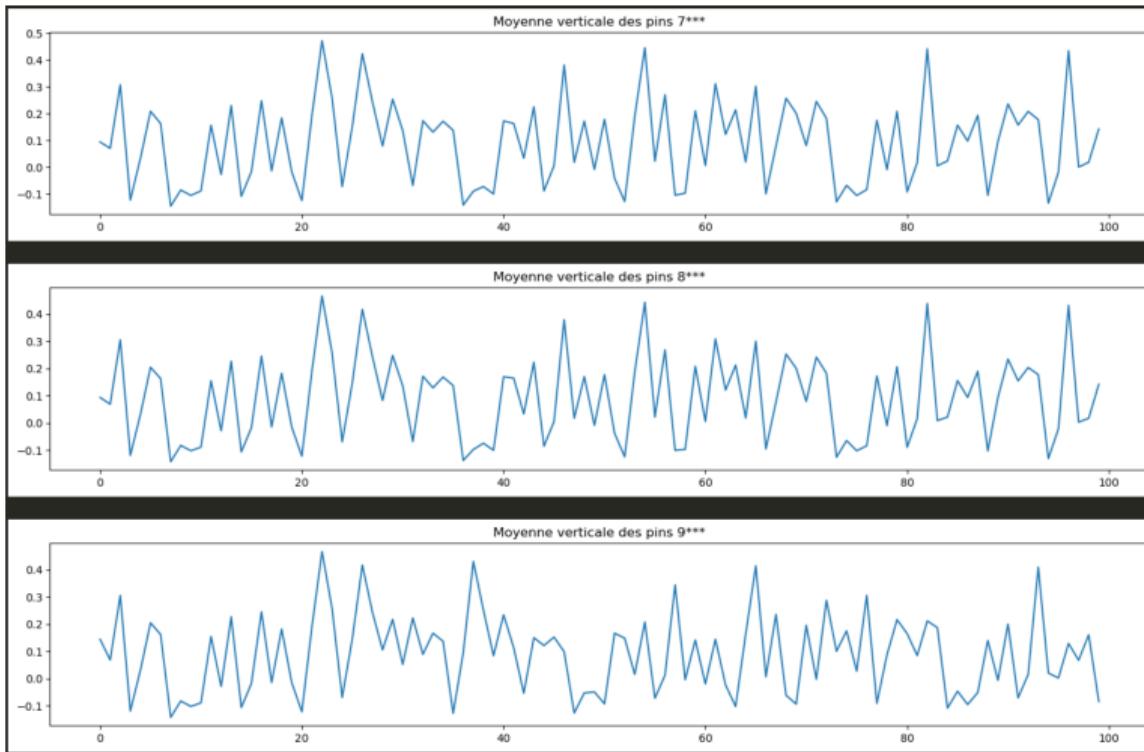
Faillock et bypass



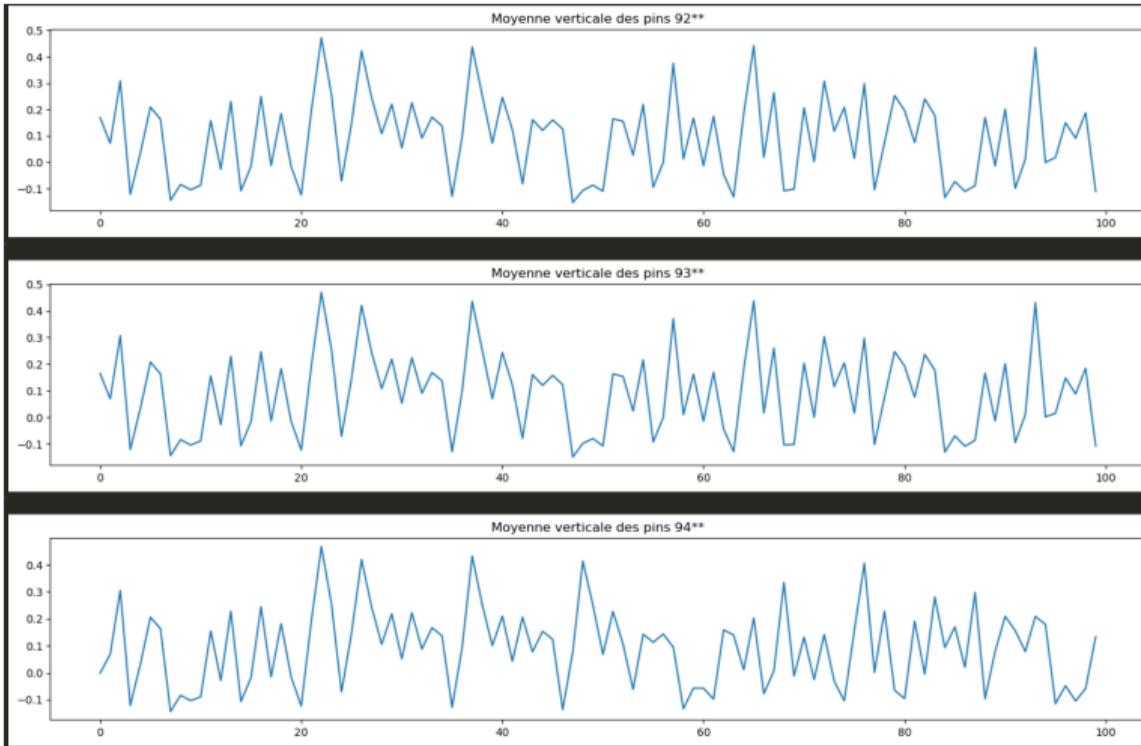
Faillock et bypass



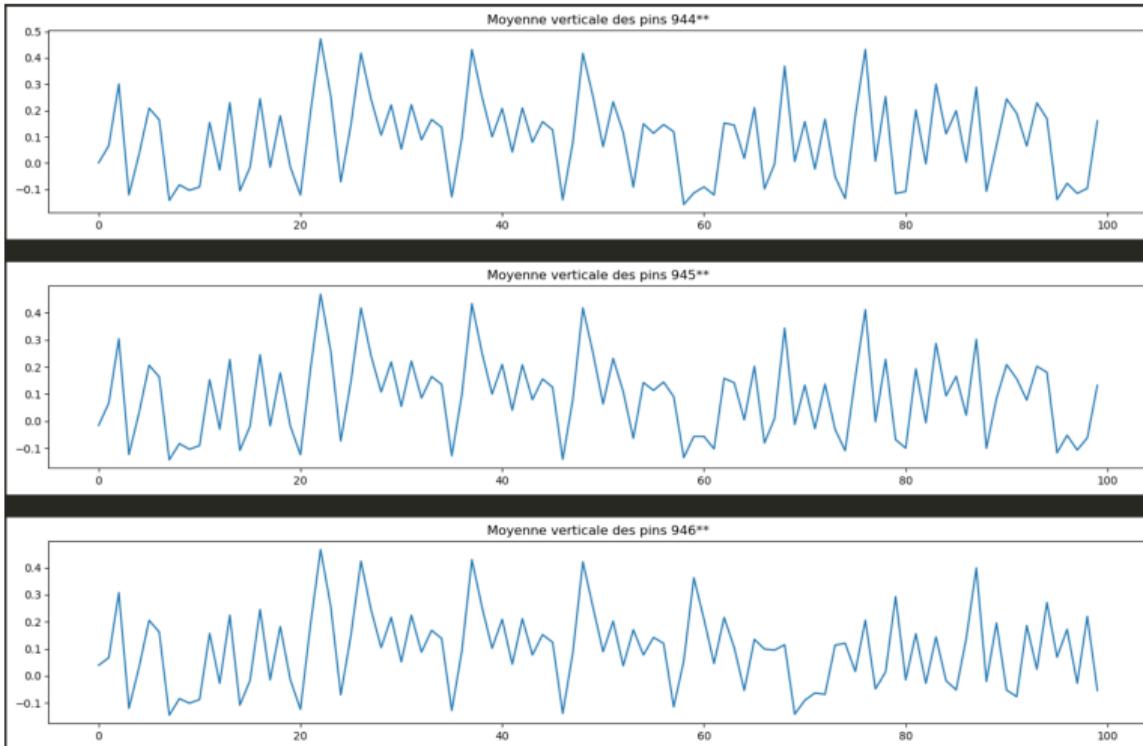
Comparaison séquentielle



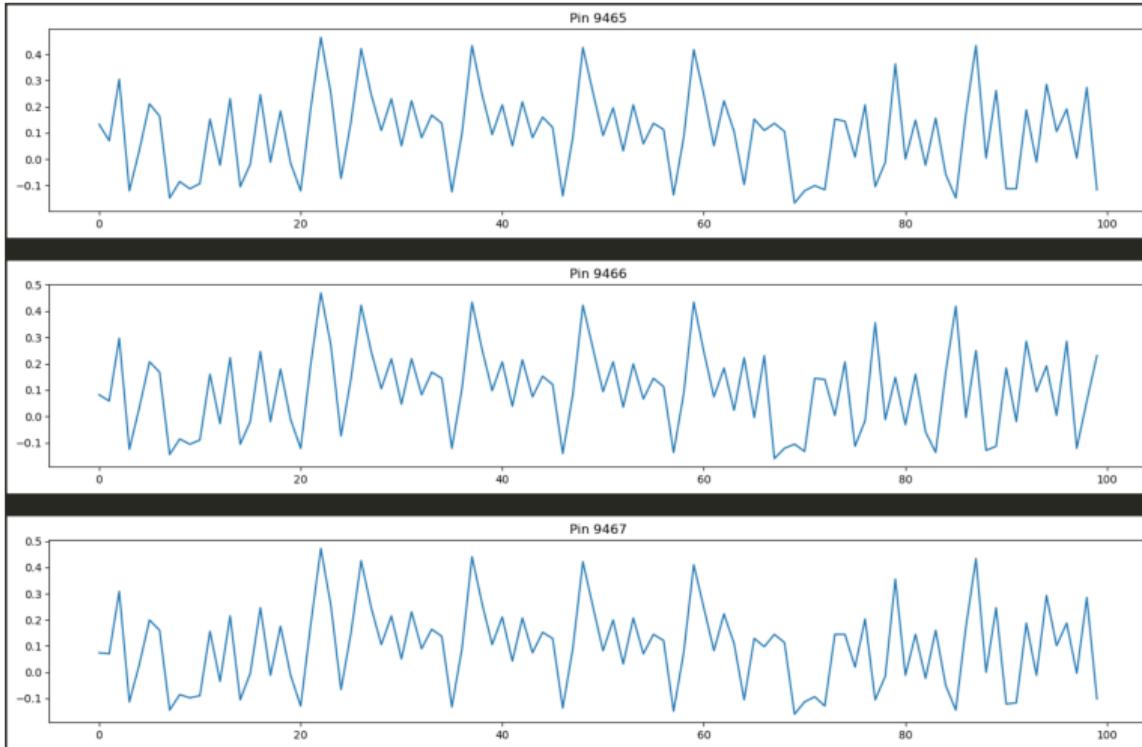
Comparaison séquentielle



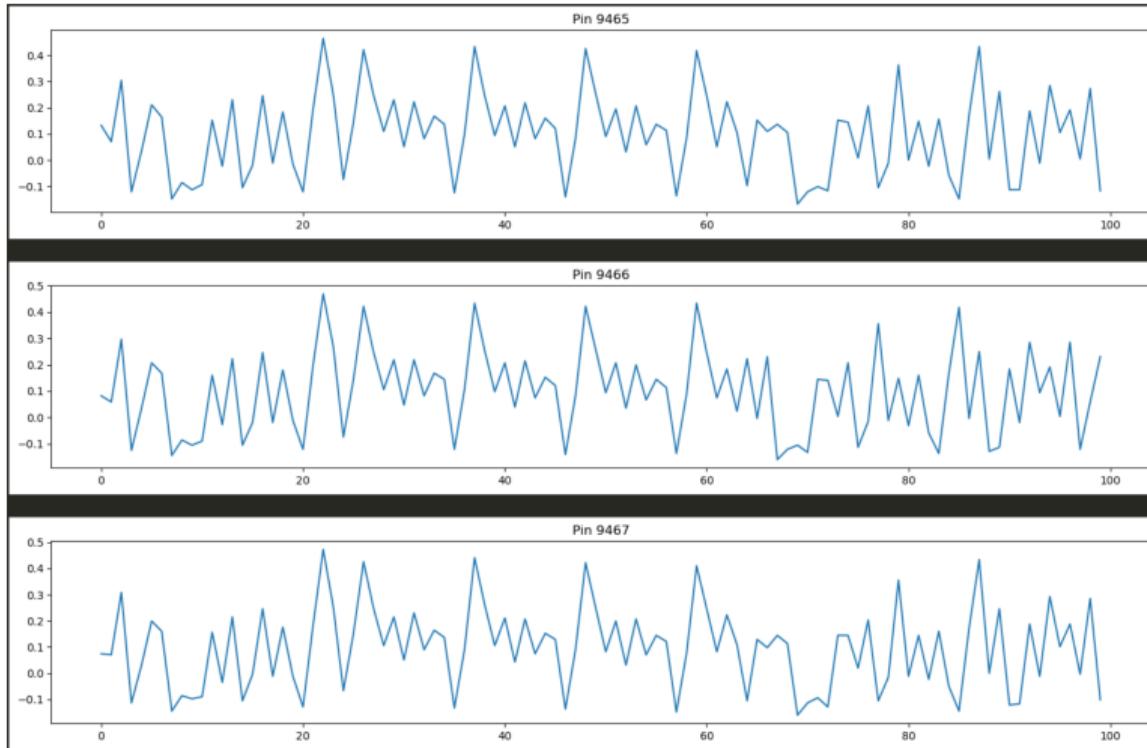
Comparaison séquentielle



Comparaison séquentielle



Comparaison séquentielle



Attaques par faute sur RSA et ECDSA

Bellcore et LLL

No Divide just Conquer ★/★★/★★★ 60/21/6 résolutions



- Objectif
 - Implémenter RSA en assembleur-like
 - Avec de plus en plus de restrictions

Atomic Secable ★★★

4 résolutions



- Objectif
 - Récupérer la clef secrète ECDSA

Atomic Secable ★★★

4 résolutions



- Objectif
 - Récupérer la clef secrète ECDSA
- Données
 - 65 536 signatures fautées avec la clef



- Objectif
 - Récupérer la clef secrète ECDSA
- Données
 - 65 536 signatures fautées avec la clef
- Capacités d'attaquant
 - Fauter aléatoirement une liste d'instruction donnée

Attaque à information partielle sur ECDSA¹

$$\left\{ \begin{array}{l} s_1 = k_1^{-1} (h_1 + r_1 * d) \mod n \\ s_2 = k_2^{-1} (h_2 + r_2 * d) \mod n \\ \vdots \\ s_m = k_m^{-1} (h_m + r_m * d) \mod n \end{array} \right.$$

Attaque à information partielle sur ECDSA¹

$$\left\{ \begin{array}{lcl} s_1 & = & k_1^{-1} (h_1 + r_1 * d) \mod n \\ s_2 & = & k_2^{-1} (h_2 + r_2 * d) \mod n \\ \vdots & & \\ s_m & = & k_m^{-1} (h_m + r_m * d) \mod n \end{array} \right. \Rightarrow \left\{ \begin{array}{lcl} k_1 + t_1 k_m + u_1 & = & 0 \mod n \\ k_2 + t_2 k_m + u_2 & = & 0 \mod n \\ \vdots & & \\ k_{m-1} + t_{m-1} k_m + u_{m-1} & = & 0 \mod n \end{array} \right.$$

Attaque à information partielle sur ECDSA¹

$$\left\{ \begin{array}{lcl} s_1 & = & k_1^{-1} (h_1 + r_1 * d) \mod n \\ s_2 & = & k_2^{-1} (h_2 + r_2 * d) \mod n \\ \vdots & & \\ s_m & = & k_m^{-1} (h_m + r_m * d) \mod n \end{array} \right. \Rightarrow \left\{ \begin{array}{lcl} k_1 + t_1 k_m + u_1 & = & 0 \mod n \\ k_2 + t_2 k_m + u_2 & = & 0 \mod n \\ \vdots & & \\ k_{m-1} + t_{m-1} k_m + u_{m-1} & = & 0 \mod n \end{array} \right.$$

On espère que les k_i soient suffisamment petits (connaître les bits de poids fort)

Attaque à information partielle sur ECDSA¹

$$\left\{ \begin{array}{lcl} s_1 & = & k_1^{-1} (h_1 + r_1 * d) \mod n \\ s_2 & = & k_2^{-1} (h_2 + r_2 * d) \mod n \\ \vdots & & \\ s_m & = & k_m^{-1} (h_m + r_m * d) \mod n \end{array} \right. \Rightarrow \left\{ \begin{array}{lcl} k_1 + t_1 k_m + u_1 & = & 0 \mod n \\ k_2 + t_2 k_m + u_2 & = & 0 \mod n \\ \vdots & & \\ k_{m-1} + t_{m-1} k_m + u_{m-1} & = & 0 \mod n \end{array} \right.$$

On espère que les k_i soient suffisamment petits (connaître les bits de poids fort)

Le réseau suivant contient le vecteur (k_1, \dots, k_m, K)

$$\left(\begin{array}{cccccc} n & & & & & \\ n & & & & & \\ \ddots & & & & & \\ & & & n & & \\ t_1 & t_2 & \dots & t_{m-1} & 1 & \\ u_1 & u_2 & \dots & u_{m-1} & 0 & K \end{array} \right)$$

Attaque à information partielle sur ECDSA¹

$$\left\{ \begin{array}{lcl} s_1 & = & k_1^{-1} (h_1 + r_1 * d) \mod n \\ s_2 & = & k_2^{-1} (h_2 + r_2 * d) \mod n \\ \vdots & & \\ s_m & = & k_m^{-1} (h_m + r_m * d) \mod n \end{array} \right. \Rightarrow \left\{ \begin{array}{lcl} k_1 + t_1 k_m + u_1 & = & 0 \mod n \\ k_2 + t_2 k_m + u_2 & = & 0 \mod n \\ \vdots & & \\ k_{m-1} + t_{m-1} k_m + u_{m-1} & = & 0 \mod n \end{array} \right.$$

On espère que les k_i soient suffisamment petits (connaître les bits de poids fort)

Le réseau suivant contient le vecteur (k_1, \dots, k_m, K)

$$\left(\begin{array}{cccccc} n & & & & & \\ & n & & & & \\ & & \ddots & & & \\ & & & n & & \\ t_1 & t_2 & \dots & t_{m-1} & 1 & \\ u_1 & u_2 & \dots & u_{m-1} & 0 & K \end{array} \right)$$

- Tricks sur l'algorithme LLL

Attaque à information partielle sur ECDSA¹

$$\left\{ \begin{array}{lcl} s_1 & = & k_1^{-1} (h_1 + r_1 * d) \mod n \\ s_2 & = & k_2^{-1} (h_2 + r_2 * d) \mod n \\ \vdots & & \\ s_m & = & k_m^{-1} (h_m + r_m * d) \mod n \end{array} \right. \Rightarrow \left\{ \begin{array}{lcl} k_1 + t_1 k_m + u_1 & = & 0 \mod n \\ k_2 + t_2 k_m + u_2 & = & 0 \mod n \\ \vdots & & \\ k_{m-1} + t_{m-1} k_m + u_{m-1} & = & 0 \mod n \end{array} \right.$$

On espère que les k_i soient suffisamment petits (connaître les bits de poids fort)

Le réseau suivant contient le vecteur (k_1, \dots, k_m, K)

$$\left(\begin{array}{cccccc} n & & & & & \\ & n & & & & \\ & & \ddots & & & \\ & & & n & & \\ t_1 & t_2 & \dots & t_{m-1} & 1 & \\ u_1 & u_2 & \dots & u_{m-1} & 0 & K \end{array} \right)$$

- Tricks sur l'algorithme LLL
- La diagonale de n permet de simuler un réseau modulaire

Attaque à information partielle sur ECDSA¹

$$\left\{ \begin{array}{lcl} s_1 & = & k_1^{-1} (h_1 + r_1 * d) \mod n \\ s_2 & = & k_2^{-1} (h_2 + r_2 * d) \mod n \\ \vdots & & \\ s_m & = & k_m^{-1} (h_m + r_m * d) \mod n \end{array} \right. \Rightarrow \left\{ \begin{array}{lcl} k_1 + t_1 k_m + u_1 & = & 0 \mod n \\ k_2 + t_2 k_m + u_2 & = & 0 \mod n \\ \vdots & & \\ k_{m-1} + t_{m-1} k_m + u_{m-1} & = & 0 \mod n \end{array} \right.$$

On espère que les k_i soient suffisamment petits (connaître les bits de poids fort)

Le réseau suivant contient le vecteur (k_1, \dots, k_m, K)

$$\left(\begin{array}{cccccc} n & & & & & \\ & n & & & & \\ & & \ddots & & & \\ & & & n & & \\ t_1 & t_2 & \dots & t_{m-1} & 1 & \\ u_1 & u_2 & \dots & u_{m-1} & 0 & K \end{array} \right)$$

- Tricks sur l'algorithme LLL

- La diagonale de n permet de simuler un réseau modulaire
- K est choisi grand pour forcer la dernière ligne à 1

Dans le cadre du challenge

- Retrouver la clef privée: si un des k_i ou certains bits de plusieurs k_i sont connus, c'est gagné

Dans le cadre du challenge

- Retrouver la clef privée: si un des k_i ou certains bits de plusieurs k_i sont connus, c'est gagné
- Objectif : distinguer les deux cas

Input: Scalaire $k = (k_{n-1}, \dots, k_0)_2$,
Point P

Output: $Q = [k]P$
 $(R_0, R_1) \leftarrow (\mathcal{O}, P);$
for $i \leftarrow n - 1$ **to** 0 **do**

if $k_i = 0$ **then**
 | $(R_0, R_1) \leftarrow ([2]R_0, R_0 + R_1);$
end
else
 | $(R_0, R_1) \leftarrow (R_0 + R_1, [2]R_1);$
end

end

return $R_0;$

Algorithm: Echelle de Montgomery

Dans le cadre du challenge

- Retrouver la clef privée: si un des k_i ou certains bits de plusieurs k_i sont connus, c'est gagné
- Objectif : distinguer les deux cas
 - Par SCA (opérations différentes)

Input: Scalaire $k = (k_{n-1}, \dots, k_0)_2$,
Point P

Output: $Q = [k]P$
 $(R_0, R_1) \leftarrow (\mathcal{O}, P);$
for $i \leftarrow n - 1$ **to** 0 **do**

if $k_i = 0$ **then**
 | $(R_0, R_1) \leftarrow ([2]R_0, R_0 + R_1);$
end
else
 | $(R_0, R_1) \leftarrow (R_0 + R_1, [2]R_1);$
end

end

return $R_0;$

Algorithm: Echelle de Montgomery

Dans le cadre du challenge

- Retrouver la clef privée: si un des k_i ou certains bits de plusieurs k_i sont connus, c'est gagné
- Objectif : distinguer les deux cas
 - Par SCA (opérations différentes)
 - Par timing (premières opérations ignorées)
 - Temps d'exécution non constant

Input: Scalaire $k = (k_{n-1}, \dots, k_0)_2$,
Point P

Output: $Q = [k]P$
 $(R_0, R_1) \leftarrow (\mathcal{O}, P);$
for $i \leftarrow n - 1$ **to** 0 **do**

if $k_i = 0$ **then**
 | $(R_0, R_1) \leftarrow ([2]R_0, R_0 + R_1);$
end
else
 | $(R_0, R_1) \leftarrow (R_0 + R_1, [2]R_1);$
end

end

return $R_0;$

Algorithm: Echelle de Montgomery

Dans le cadre du challenge

- Retrouver la clef privée: si un des k_i ou certains bits de plusieurs k_i sont connus, c'est gagné
- Objectif : distinguer les deux cas
 - Par SCA (opérations différentes)
 - Par timing (premières opérations ignorées)
 - Temps d'exécution non constant
 - Un seul des deux cas resiste à une faute
 - On faute les 16 premières opérations
 - Les survivants commencent par 16 zéros

Input: Scalaire $k = (k_{n-1}, \dots, k_0)_2$,
Point P

Output: $Q = [k]P$
 $(R_0, R_1) \leftarrow (\mathcal{O}, P);$
for $i \leftarrow n - 1$ **to** 0 **do**

if $k_i = 0$ **then**
 | $(R_0, R_1) \leftarrow ([2]R_0, R_0 + R_1);$
end
else
 | $(R_0, R_1) \leftarrow (R_0 + R_1, [2]R_1);$
end

end

return $R_0;$

Algorithm: Echelle de Montgomery

Fuite de l'état interne AES

Dévoiler la clef, un tour à la fois



- Objectif
 - Décoder un message chiffré avec AES



- Objectif
 - Décoder un message chiffré avec AES
- Données
 - Observer l'évolution d'un registre à chaque étape du chiffrement

Capacité d'attaquant

Révéler une variable (registre) à une ligne (IP) donnée

```
1 for (int i = 0; i < 16; i++) {  
2     s[i] = s[i] ^ k[i];  
3 }
```

- "Donne moi k à la ligne 2"
 - On récupère tout k

Capacité d'attaquant

Révéler une variable (registre) à une ligne (IP) donnée

```
1 for (int i = 0; i < 16; i++) {  
2     s[i] = s[i] ^ k[i];  
3 }
```

- "Donne moi k à la ligne 2"
 - On récupère tout k
- Loop-unrolling
 - Réduit l'overhead lié aux boucles
 - Binaires plus lourd

Capacité d'attaquant

Révéler une variable (registre) à une ligne (IP) donnée

```
1 for (int i = 0; i < 16; i++) {  
2     s[i] = s[i] ^ k[i];  
3 }
```

- "Donne moi k à la ligne 2"
 - On récupère tout k
- Loop-unrolling
 - Réduit l'overhead lié aux boucles
 - Binaires plus lourd
- "Donne moi k à la ligne 2"
 - On récupère un byte de k

```
1     s[0] = s[0] ^ k[0];  
2     s[1] = s[1] ^ k[1];  
3     ...  
4     s[14] = s[14] ^ k[14];  
5     s[15] = s[15] ^ k[15];
```

Analyse du loop-unrolling (code C)

- **KeySchedule** : 1 bytes du keySchedule par tour (sauf un) 48 bits de bruteforce

Analyse du loop-unrolling (code C)

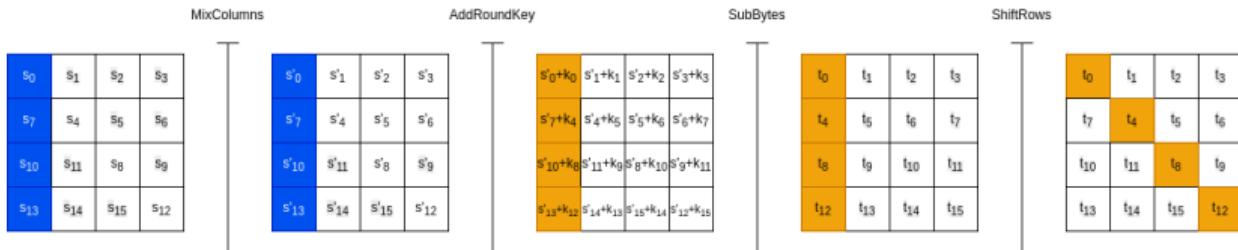
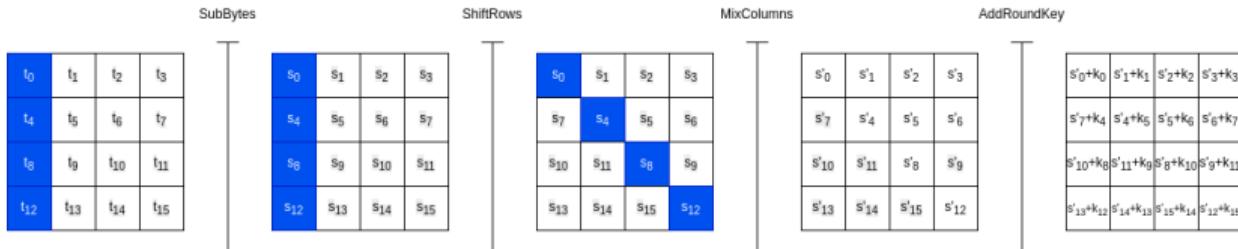
- **KeySchedule** : 1 bytes du keySchedule par tour (sauf un) 48 bits de bruteforce
- **MixColumns** : 1 byte de l'état interne par tour
- **AddRoundKey** : 1 byte de l'état interne/clef par tour 40 bits de bruteforce

Analyse du loop-unrolling (code C)

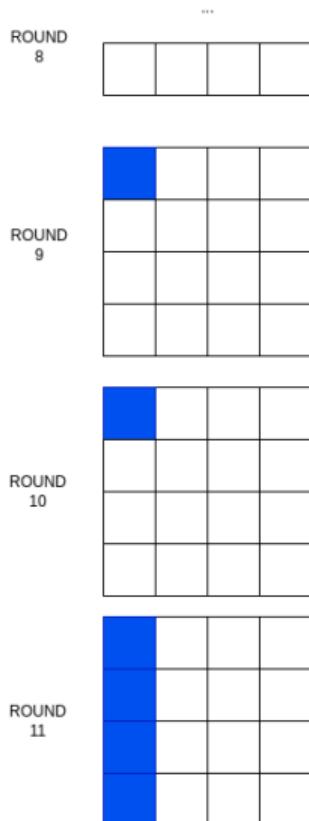
- **KeySchedule** : 1 bytes du keySchedule par tour (sauf un) 48 bits de bruteforce
- **MixColumns** : 1 byte de l'état interne par tour
- **AddRoundKey** : 1 byte de l'état interne/clef par tour 40 bits de bruteforce
- **Shiftrows + Subbytes** : 4 bytes (une colonne) de l'état interne par tour

Analyse du loop-unrolling (code C)

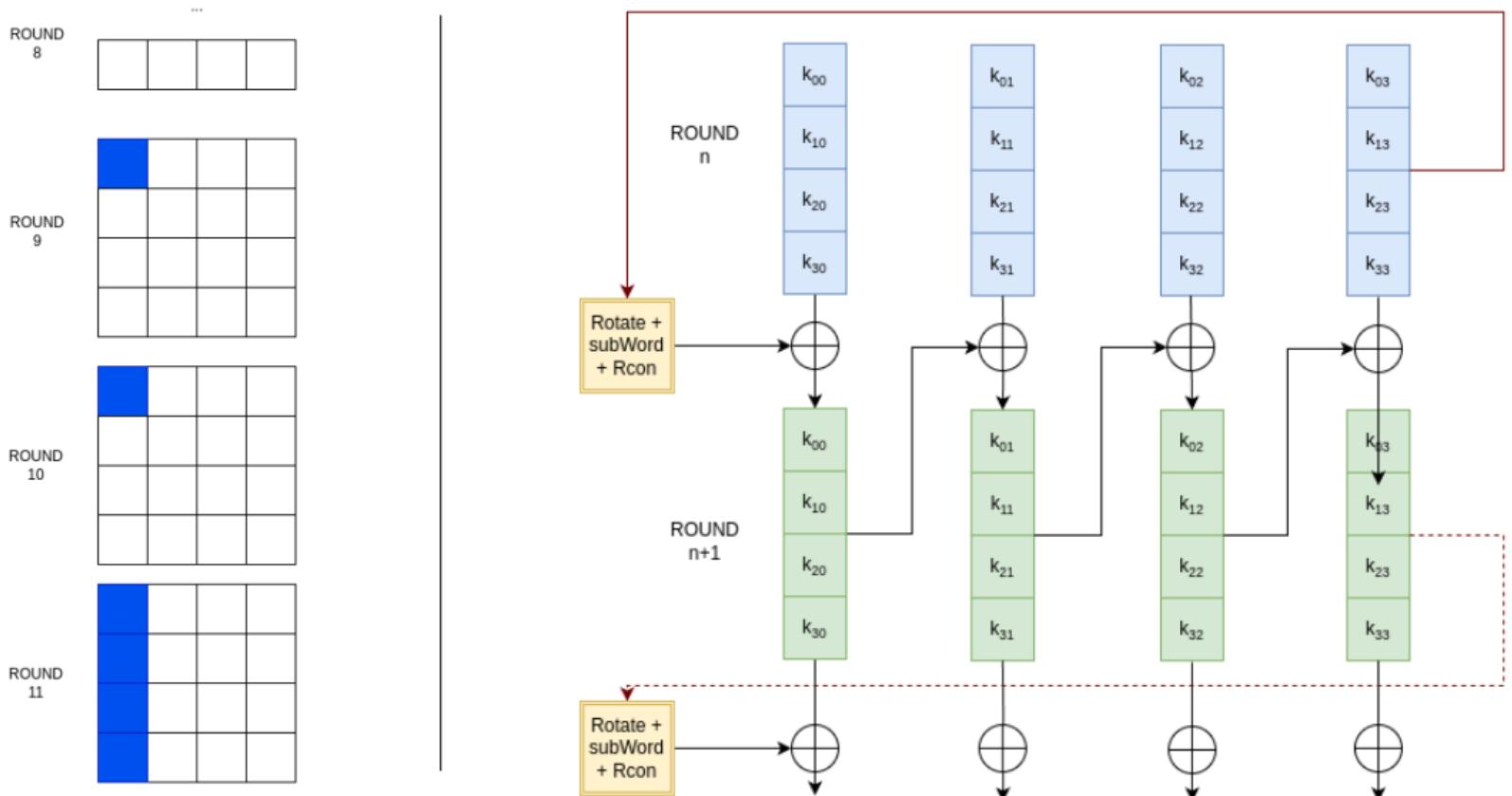
- **KeySchedule** : 1 bytes du keySchedule par tour (sauf un) 48 bits de bruteforce
- **MixColumns** : 1 byte de l'état interne par tour
- **AddRoundKey** : 1 byte de l'état interne/clef par tour 40 bits de bruteforce
- **Shiftrows + Subbytes** : 4 bytes (une colonne) de l'état interne par tour



Rappel sur le KeySchedule

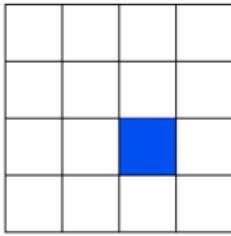
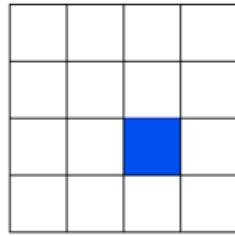


Rappel sur le KeySchedule

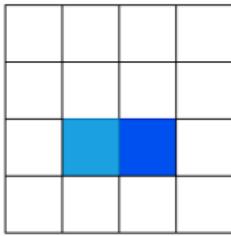
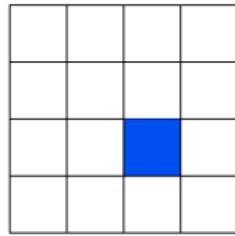


Propagation de connaissance

ROUND
 n

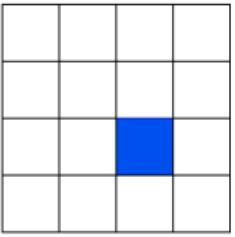
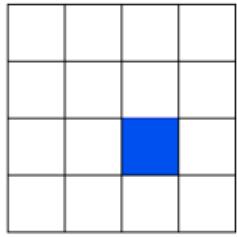


ROUND
 $n+1$

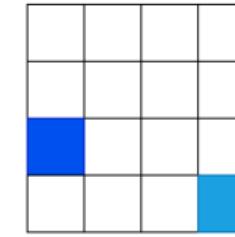
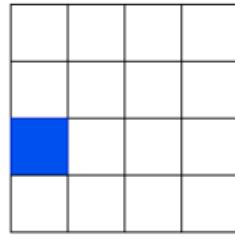


Propagation de connaissance

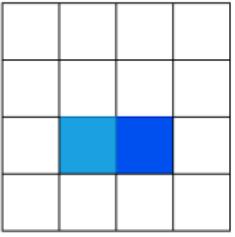
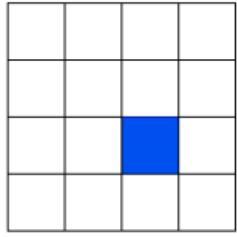
ROUND
 n



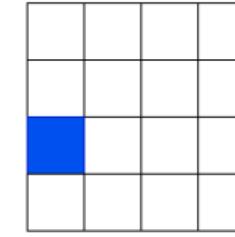
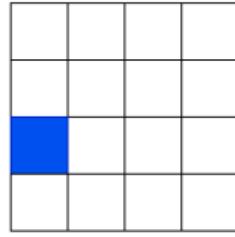
ROUND
 n



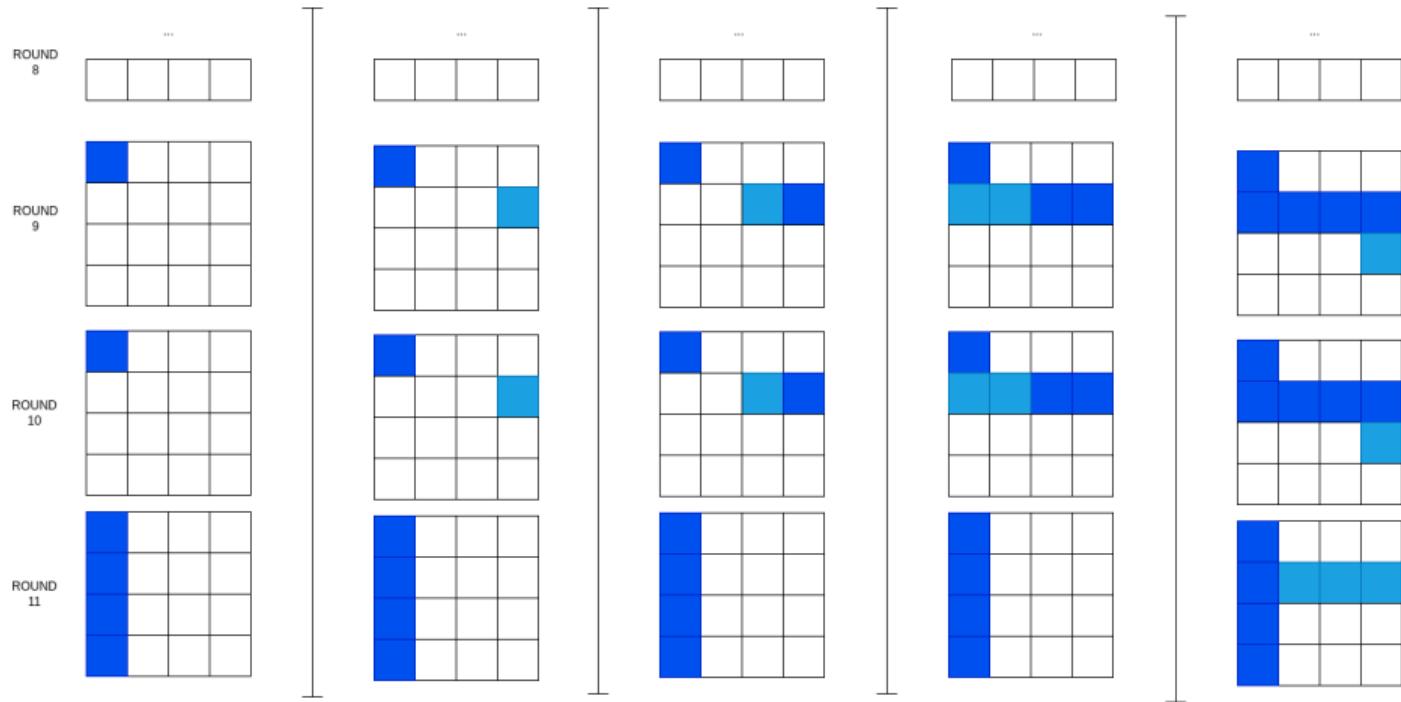
ROUND
 $n+1$



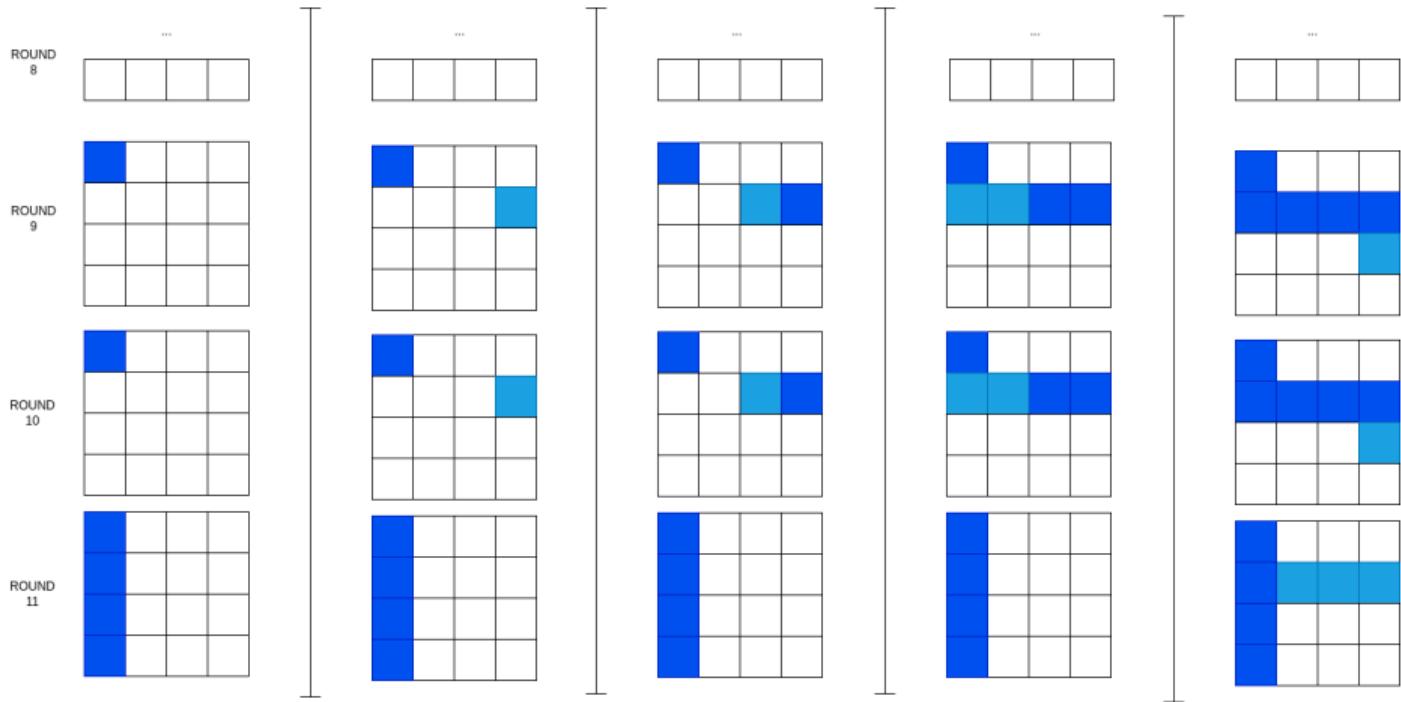
ROUND
 $n+1$



Full recovery



Full recovery



Au final, il reste 3 bytes à bruteforcer (24 bits) 

Introduction : Les classiques de ctf

Cinq challenges de démarrage



- Objectif
 - Récupérer d aléatoire sur 666 bits



- Objectif
 - Récupérer d aléatoire sur 666 bits
- Données
 - $d_p = d^{-1} \pmod{p-1}$
 - $d_q = d^{-1} \pmod{q-1}$



- Objectif
 - Récupérer d aléatoire sur 666 bits
- Données
 - $d_p = d^{-1} \pmod{p-1}$
 - $d_q = d^{-1} \pmod{q-1}$
 - p, q sur 512 bits



- Objectif
 - Récupérer d aléatoire sur 666 bits
- Données
 - $d_p = d^{-1} \pmod{p-1}$
 - $d_q = d^{-1} \pmod{q-1}$
 - p, q sur 512 bits
- Solution
 - Théorème des restes chinois pour avoir
$$d \pmod{\text{lcm}(p-1, q-1)}$$



- Objectif
 - Récupérer d aléatoire sur 666 bits
- Données
 - $d_p = d^{-1} \pmod{p-1}$
 - $d_q = d^{-1} \pmod{q-1}$
 - p, q sur 512 bits
- Solution
 - Théorème des restes chinois pour avoir
$$d \pmod{\text{lcm}(p-1, q-1)}$$
- First Blood en 65 secondes 



- Objectif
 - Forger un chiffrement + authentification
- Données
 - Accès à un oracle
- Solution
 - Malléabilité d'AES-CTR / AES-OFB
- First Blood en 7 minutes ━



- Objectif
 - Résoudre un système d'équation diophantienne

$$\begin{cases} a &= 487c \\ 59a &= 485b \\ x^2 &= a + b \\ y(3y - 1) &= 2b \end{cases}$$

- Solution
 - Se ramener à l'équation de Pell :
$$(6y - 1)^2 - 145710941544k^2 = 1$$
- Peut se résoudre avec des fractions continues
- <https://www.alpertron.com.ar/METHODS.HTM>





- Objectif
 - Résoudre des équations modulo s aléatoire
 - $E : m \mapsto IV \cdot m \pmod{s} = c$
 - $E^{-1} : c \mapsto IV^{-1} \cdot c \pmod{s} = m$



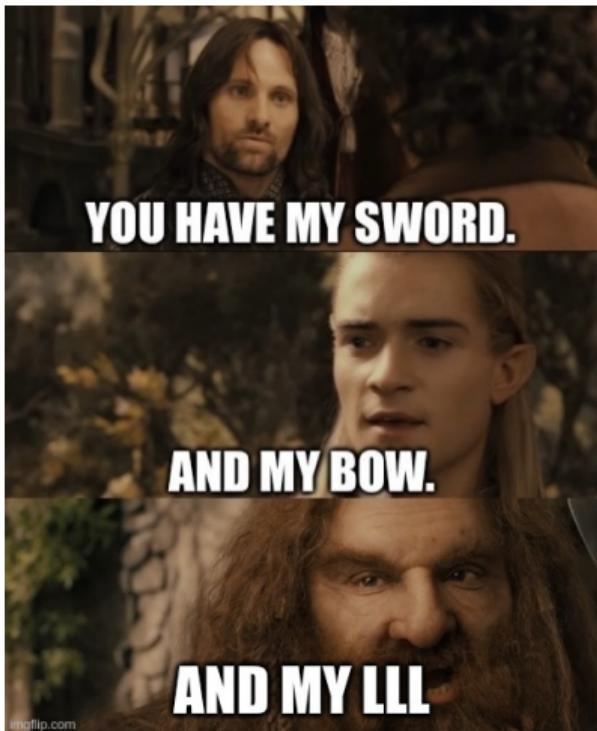
- Objectif
 - Résoudre des équations modulo s aléatoire
 - $E : m \mapsto IV \cdot m \pmod{s} = c$
 - $E^{-1} : c \mapsto IV^{-1} \cdot c \pmod{s} = m$
- Données
 - Deux clairs connus $E(m_1) = c_1$ et $E(m_2) = c_2$
 - Un message à déchiffrer m_3 sachant c_3



- Objectif
 - Résoudre des équations modulo s aléatoire
 - $E : m \mapsto IV \cdot m \pmod{s} = c$
 - $E^{-1} : c \mapsto IV^{-1} \cdot c \pmod{s} = m$
- Données
 - Deux clairs connus $E(m_1) = c_1$ et $E(m_2) = c_2$
 - Un message à déchiffrer m_3 sachant c_3
- Solution
 - $$\begin{cases} c_1 &= IV_1 \cdot m_1 + k_1 s \\ c_2 &= IV_2 \cdot m_2 + k_2 s \end{cases}$$



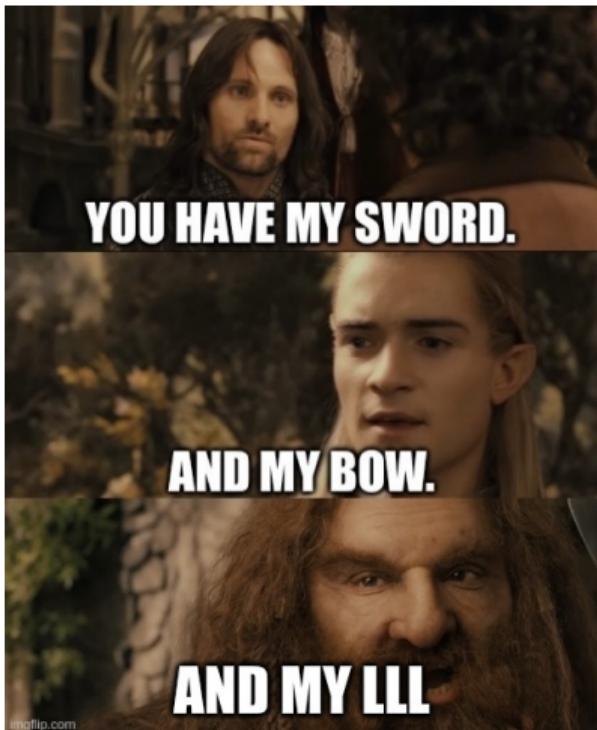
- Objectif
 - Résoudre des équations modulo s aléatoire
 - $E : m \mapsto IV \cdot m \pmod{s} = c$
 - $E^{-1} : c \mapsto IV^{-1} \cdot c \pmod{s} = m$
- Données
 - Deux clairs connus $E(m_1) = c_1$ et $E(m_2) = c_2$
 - Un message à déchiffrer m_3 sachant c_3
- Solution
 - $$\begin{cases} c_1 &= IV_1 \cdot m_1 + k_1 s \\ c_2 &= IV_2 \cdot m_2 + k_2 s \end{cases}$$
 - $s = PGCD(c_1 - IV_1 \cdot m_1, c_2 - IV_2 \cdot m_2)$ ━



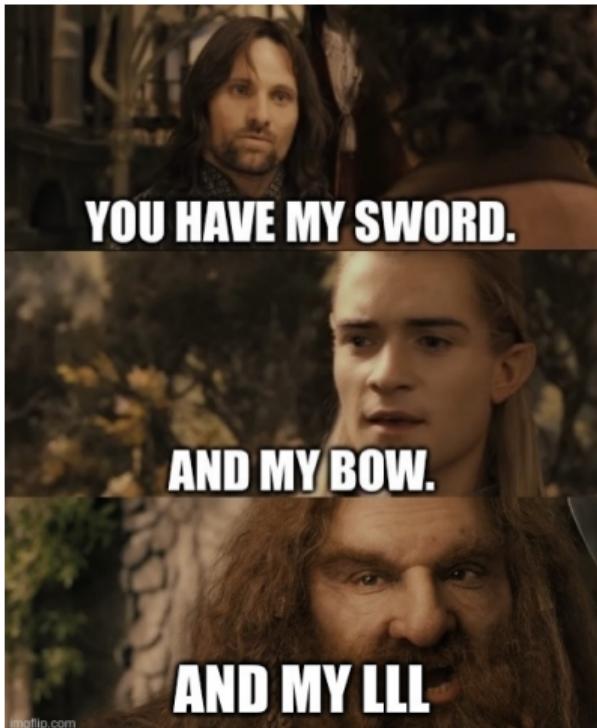
- Objectif
 - Résoudre des équations modulo s aléatoire
 - $E : m \mapsto IV \cdot m \pmod{s} = c$
 - $E^{-1} : c \mapsto IV^{-1} \cdot c \pmod{s} = m$
- Données
 - Les chiffrés : $E(m_1) = c_1$ et $E(m_2) = c_2$



- Objectif
 - Résoudre des équations modulo s aléatoire
 - $E : m \mapsto IV \cdot m \pmod{s} = c$
 - $E^{-1} : c \mapsto IV^{-1} \cdot c \pmod{s} = m$
- Données
 - Les chiffrés : $E(m_1) = c_1$ et $E(m_2) = c_2$
- Solution
 - $$\begin{cases} c_1 = IV_1 \cdot m_1 + k_1 s \\ c_2 = IV_2 \cdot m_2 + k_2 s \end{cases}$$
 - $c_1 k_2 - IV_1 m_1 k_2 - c_2 k_1 + IV_2 m_2 k_1 = 0$



- Objectif
 - Résoudre des équations modulo s aléatoire
 - $E : m \mapsto IV \cdot m \pmod{s} = c$
 - $E^{-1} : c \mapsto IV^{-1} \cdot c \pmod{s} = m$
- Données
 - Les chiffrés : $E(m_1) = c_1$ et $E(m_2) = c_2$
- Solution
 - $$\begin{cases} c_1 = IV_1 \cdot m_1 + k_1 s \\ c_2 = IV_2 \cdot m_2 + k_2 s \end{cases}$$
 - $c_1 k_2 - IV_1 m_1 k_2 - c_2 k_1 + IV_2 m_2 k_1 = 0$



- Objectif
 - Résoudre des équations modulo s aléatoire
 - $E : m \mapsto IV \cdot m \pmod{s} = c$
 - $E^{-1} : c \mapsto IV^{-1} \cdot c \pmod{s} = m$
- Données
 - Les chiffrés : $E(m_1) = c_1$ et $E(m_2) = c_2$
- Solution
 - $$\begin{cases} c_1 = IV_1 \cdot m_1 + k_1 s \\ c_2 = IV_2 \cdot m_2 + k_2 s \end{cases}$$
 - $c_1 k_2 - IV_1 m_1 k_2 - c_2 k_1 + IV_2 m_2 k_1 = 0$
 - s et IV_i font 1024 bits, m_i font 256 bits
- Résoudre avec Coppersmith ou LLL ?

Coppersmith / LLL

Utilisation de l'algorithme LLL en ctf

Trouve des combinaisons linéaires anormalement petites de vecteurs.

Coppersmith / LLL

Utilisation de l'algorithme LLL en ctf

Trouve des combinaisons linéaires anormalement petites de vecteurs.

$$\begin{array}{cccc} 1024 & 1024 & 1024 & 1024 \text{ bits} \\ \downarrow & \downarrow & \downarrow & \downarrow \\ c_1 k_2 - IV_1 m_1 k_2 - c_2 k_1 + IV_2 m_2 k_1 = 0 \end{array}$$

Coppersmith / LLL

Utilisation de l'algorithme LLL en ctf

Trouve des combinaisons linéaires anormalement petites de vecteurs.

$$\begin{array}{cccc} 1024 & 1024 & 1024 & 1024 \text{ bits} \\ \downarrow & \downarrow & \downarrow & \downarrow \\ c_1 k_2 - IV_1 m_1 k_2 - c_2 k_1 + IV_2 m_2 k_1 = 0 \end{array}$$

- 0 est engendré par (c_1) , (IV_1) , (c_2) , (IV_2)

Coppersmith / LLL

Utilisation de l'algorithme LLL en ctf

Trouve des combinaisons linéaires anormalement petites de vecteurs.

$$\begin{array}{cccc} 1024 & 1024 & 1024 & 1024 \text{ bits} \\ \downarrow & \downarrow & \downarrow & \downarrow \\ c_1 k_2 - IV_1 m_1 k_2 - c_2 k_1 + IV_2 m_2 k_1 = 0 \end{array}$$

- 0 est engendré par (c_1) , (IV_1) , (c_2) , (IV_2)
→ LLL retourne bien 0
- Comment retrouver les coefficients utilisés?

Coppersmith / LLL

Utilisation de l'algorithme LLL en ctf

Trouve des combinaisons linéaires anormalement petites de vecteurs.

$$\begin{array}{cccc} 1024 & 1024 & 1024 & 1024 \text{ bits} \\ \downarrow & \downarrow & \downarrow & \downarrow \\ c_1 k_2 - IV_1 m_1 k_2 - c_2 k_1 + IV_2 m_2 k_1 = 0 \end{array}$$

- 0 est engendré par (c_1) , (IV_1) , (c_2) , (IV_2)
→ LLL retourne bien 0
- Comment retrouver les coefficients utilisés?
- $\begin{bmatrix} c_1, & 1, & 0, & 0, & 0 \\ -IV_1, & 0, & 1, & 0, & 0 \\ -c_2, & 0, & 0, & 1, & 0 \\ IV_2, & 0, & 0, & 0, & 1 \end{bmatrix}$ engendre le vecteur $(0, k_2, m_1 k_2, k_1, m_2 k_1)$

Coppersmith / LLL

Utilisation de l'algorithme LLL en ctf

Trouve des combinaisons linéaires anormalement petites de vecteurs.

$$\begin{array}{cccc} 1024 & 1024 & 1024 & 1024 \text{ bits} \\ \downarrow & \downarrow & \downarrow & \downarrow \\ c_1 k_2 - IV_1 m_1 k_2 - c_2 k_1 + IV_2 m_2 k_1 = 0 \end{array}$$

- 0 est engendré par (c_1) , (IV_1) , (c_2) , (IV_2)
→ LLL retourne bien 0
- Comment retrouver les coefficients utilisés?
- $\begin{bmatrix} c_1, & 1, & 0, & 0, & 0 \\ -IV_1, & 0, & 1, & 0, & 0 \\ -c_2, & 0, & 0, & 1, & 0 \\ IV_2, & 0, & 0, & 0, & 1 \end{bmatrix}$ engendre le vecteur $(0, k_2, m_1 k_2, k_1, m_2 k_1)$
- Il a pour taille $(1, 256, 512, 256, 512)$ bits. Est-il anormalement petit?

Coppersmith / LLL

Utilisation de l'algorithme LLL en ctf

Trouve des combinaisons linéaires anormalement petites de vecteurs.

$$\begin{array}{cccc} 1024 & 1024 & 1024 & 1024 \text{ bits} \\ \downarrow & \downarrow & \downarrow & \downarrow \\ c_1 k_2 - IV_1 m_1 k_2 - c_2 k_1 + IV_2 m_2 k_1 = 0 \end{array}$$

- 0 est engendré par (c_1) , (IV_1) , (c_2) , (IV_2)
→ LLL retourne bien 0
- Comment retrouver les coefficients utilisés?
- $\begin{bmatrix} c_1, & 1, & 0, & 0, & 0 \\ -IV_1, & 0, & 1, & 0, & 0 \\ -c_2, & 0, & 0, & 1, & 0 \\ IV_2, & 0, & 0, & 0, & 1 \end{bmatrix}$ engendre le vecteur $(0, k_2, m_1 k_2, k_1, m_2 k_1)$
- Il a pour taille $(1, 256, 512, 256, 512)$ bits. Est-il anormalement petit?
→ LLL trouve un vecteur de $(254, 251, 256, 255, 257)$ bits

Changer la géométrie d'un réseau

- $\begin{bmatrix} \textcolor{blue}{c}_1 \times \mathbf{B}, & 1, & 0, & 0, & 0 \\ -\textcolor{blue}{IV}_1 \times \mathbf{B}, & 0, & 1, & 0, & 0 \\ -\textcolor{blue}{c}_2 \times \mathbf{B}, & 0, & 0, & 1, & 0 \\ \textcolor{blue}{IV}_2 \times \mathbf{B}, & 0, & 0, & 0, & 1 \end{bmatrix}$ engendre toujours $(0, \textcolor{red}{k}_2, \textcolor{red}{m}_1 \textcolor{red}{k}_2, \textcolor{red}{k}_1, \textcolor{red}{m}_2 \textcolor{red}{k}_1)$

Changer la géométrie d'un réseau

- $\begin{bmatrix} \textcolor{blue}{c}_1 \times \mathbf{B}, & 1, & 0, & 0, & 0 \\ -\textcolor{blue}{IV}_1 \times \mathbf{B}, & 0, & 1, & 0, & 0 \\ -\textcolor{blue}{c}_2 \times \mathbf{B}, & 0, & 0, & 1, & 0 \\ \textcolor{blue}{IV}_2 \times \mathbf{B}, & 0, & 0, & 0, & 1 \end{bmatrix}$ engendre toujours $(0, \textcolor{red}{k}_2, \textcolor{red}{m}_1 \textcolor{red}{k}_2, \textcolor{red}{k}_1, \textcolor{red}{m}_2 \textcolor{red}{k}_1)$
- Un \mathbf{B} gigantesque forcera la première composante à 0
→ LLL trouve un vecteur de taille $(1, 339, 339, 340, 341)$ bits

Changer la géométrie d'un réseau

- $\begin{bmatrix} \mathbf{c}_1 \times \mathbf{B}, & 1, & 0, & 0, & 0 \\ -\mathbf{IV}_1 \times \mathbf{B}, & 0, & 1, & 0, & 0 \\ -\mathbf{c}_2 \times \mathbf{B}, & 0, & 0, & 1, & 0 \\ \mathbf{IV}_2 \times \mathbf{B}, & 0, & 0, & 0, & 1 \end{bmatrix}$ engendre toujours $(0, k_2, m_1 k_2, k_1, m_2 k_1)$
- Un \mathbf{B} gigantesque forcera la première composante à 0
→ LLL trouve un vecteur de taille $(1, 339, 339, 340, 341)$ bits
- $\begin{bmatrix} \mathbf{c}_1 \times \mathbf{B}, & 2^{256}, & 0, & 0, & 0 \\ -\mathbf{IV}_1 \times \mathbf{B}, & 0, & 1, & 0, & 0 \\ -\mathbf{c}_2 \times \mathbf{B}, & 0, & 0, & 2^{256}, & 0 \\ \mathbf{IV}_2 \times \mathbf{B}, & 0, & 0, & 0, & 1 \end{bmatrix}$ engendre $(0, 2^{256} k_2, m_1 k_2, 2^{256} k_1, m_2 k_1)$

Changer la géométrie d'un réseau

- $\begin{bmatrix} \mathbf{c}_1 \times \mathbf{B}, & 1, & 0, & 0, & 0 \\ -\mathbf{IV}_1 \times \mathbf{B}, & 0, & 1, & 0, & 0 \\ -\mathbf{c}_2 \times \mathbf{B}, & 0, & 0, & 1, & 0 \\ \mathbf{IV}_2 \times \mathbf{B}, & 0, & 0, & 0, & 1 \end{bmatrix}$ engendre toujours $(0, k_2, m_1 k_2, k_1, m_2 k_1)$
- Un \mathbf{B} gigantesque forcera la première composante à 0
→ LLL trouve un vecteur de taille $(1, 339, 339, 340, 341)$ bits
- $\begin{bmatrix} \mathbf{c}_1 \times \mathbf{B}, & 2^{256}, & 0, & 0, & 0 \\ -\mathbf{IV}_1 \times \mathbf{B}, & 0, & 1, & 0, & 0 \\ -\mathbf{c}_2 \times \mathbf{B}, & 0, & 0, & 2^{256}, & 0 \\ \mathbf{IV}_2 \times \mathbf{B}, & 0, & 0, & 0, & 1 \end{bmatrix}$ engendre $(0, 2^{256} k_2, m_1 k_2, 2^{256} k_1, m_2 k_1)$
→ C'est bien le vecteur trouvé par LLL 

Sur les fonctions de hashage md5 et sha1

Comment miner du bitcoin pour moins cher?

COLLISIONS SUR MD5



COLLISIONS SUR SHA1

imgflip.com

- Objectifs : Générer un message m tel que
 - m commence par un challenge
 - $\text{sha1}(m)$ termine par les bytes 0xFC5C25
 - $\text{md5}(m)$ termine par les bytes 0xFC5C25
 - En moins de 30 minutes

Comment générer des collisions

- Brute force
 - Générer m jusqu'à avoir les trois bytes de $md5(m)$ (une chance sur 2^{24})

²[woon Kim et al., 2012]

Comment générer des collisions

- Brute force
 - Générer m jusqu'à avoir les trois bytes de $md5(m)$ (une chance sur 2^{24})
 - Prier pour les trois bytes de $sha1(m)$ (une chance sur 2^{24})

²[woon Kim et al., 2012]

Comment générer des collisions

- Brute force
 - Générer m jusqu'à avoir les trois bytes de $md5(m)$ (une chance sur 2^{24})
 - Prier pour les trois bytes de $sha1(m)$ (une chance sur 2^{24})
→ Temps total $O(2^{48})$

²[woon Kim et al., 2012]

Comment générer des collisions

- Brute force
 - Générer m jusqu'à avoir les trois bytes de $md5(m)$ (une chance sur 2^{24})
 - Prier pour les trois bytes de $sha1(m)$ (une chance sur 2^{24})
→ Temps total $O(2^{48})$
- Pay2win
 - 48 bits de collision nécessaire en 30 minutes → 155 GH/s (Giga hashes par seconde)

Hardware	Hashs (GH/s)	Prix
NVIDIA GeForce RTX 4090 (SHA1)	52	2 174€
NVIDIA GeForce RTX 5090 (SHA1)	71	2 300€
FPGAs / ASIC (SHA1)	7.3 ²	1 300€

Table: Quelques benchmarks de calcul de hashs, à la louche

²[woon Kim et al., 2012]

Comment générer des collisions

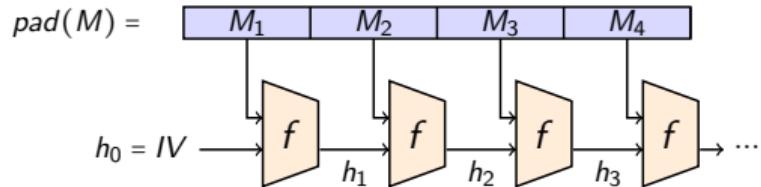
- Brute force
 - Générer m jusqu'à avoir les trois bytes de $md5(m)$ (une chance sur 2^{24})
 - Prier pour les trois bytes de $sha1(m)$ (une chance sur 2^{24})
→ Temps total $O(2^{48})$
- Pay2win
 - 48 bits de collision nécessaire en 30 minutes → 155 GH/s (Giga hashes par seconde)

Hardware	Hashs (GH/s)	Prix
NVIDIA GeForce RTX 4090 (SHA1)	52	2 174€
NVIDIA GeForce RTX 5090 (SHA1)	71	2 300€
FPGAs / ASIC (SHA1)	7.3 ²	1 300€

- Cryptanalyse
 - md5 : Collision en quelque secondes [Stevens, 2006]
 - sha1 : Collision en quelques années GPUs [Leurent and Peyrin, 2020]

²[woon Kim et al., 2012]

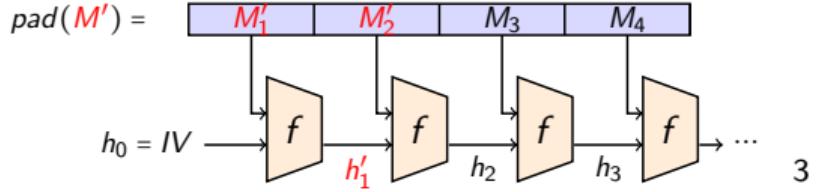
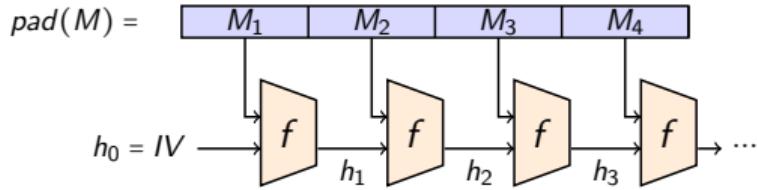
Construction de Merkle-Damgård et collisions



³[Jean, 2016]

⁴<https://github.com/cr-marcstevens/hashclash>

Construction de Merkle-Damgård et collisions



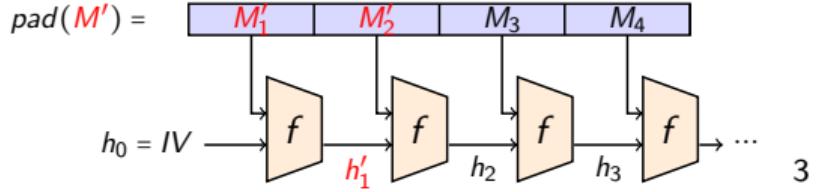
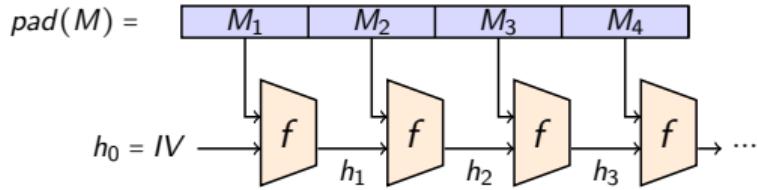
- Une collision pour h_2 entraînera une collision sur tous les états suivants
→ Si $md5(m) = md5(m')$, alors $md5(m||a) = md5(m'||a)$

3

³[Jean, 2016]

⁴<https://github.com/cr-marcstevens/hashclash>

Construction de Merkle-Damgård et collisions



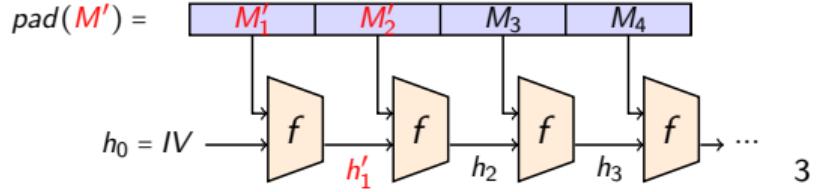
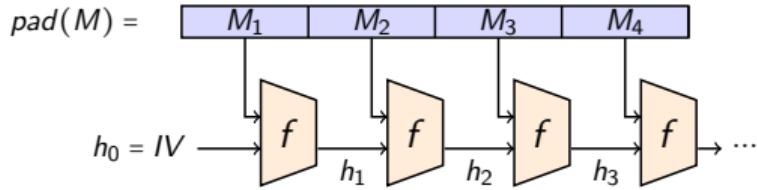
3

- Une collision pour h_2 entraînera une collision sur tous les états suivants
→ Si $md5(m) = md5(m')$, alors $md5(m||a) = md5(m'||a)$
- Dans notre cas :
 - Collision md5 pour m, m' en quelques secondes ⁴

³[Jean, 2016]

⁴<https://github.com/cr-marcstevens/hashclash>

Construction de Merkle-Damgård et collisions



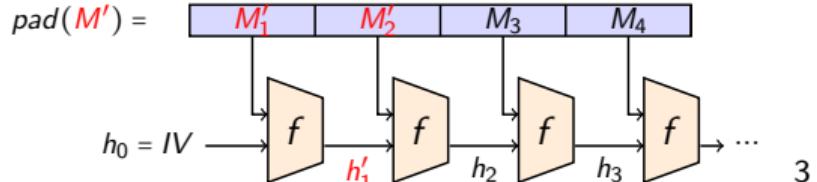
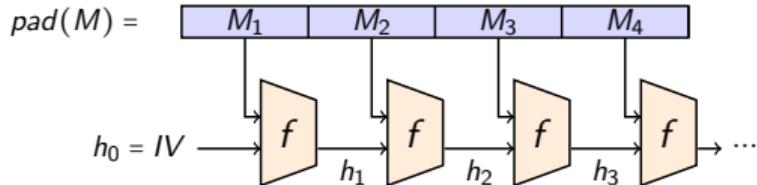
3

- Une collision pour h_2 entraînera une collision sur tous les états suivants
→ Si $md5(m) = md5(m')$, alors $md5(m||a) = md5(m'||a)$
- Dans notre cas :
 - Collision md5 pour m, m' en quelques secondes ⁴
 - 2^{24} candidats a pour avoir les 3 derniers bytes de $md5(m||a)$

³[Jean, 2016]

⁴<https://github.com/cr-marcstevens/hashclash>

Construction de Merkle-Damgård et collisions



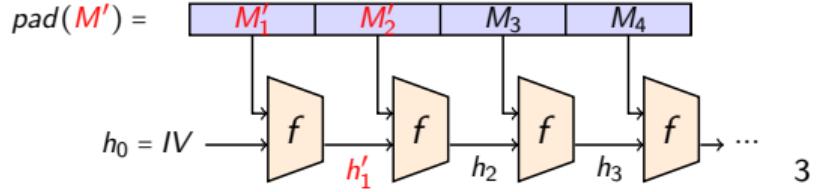
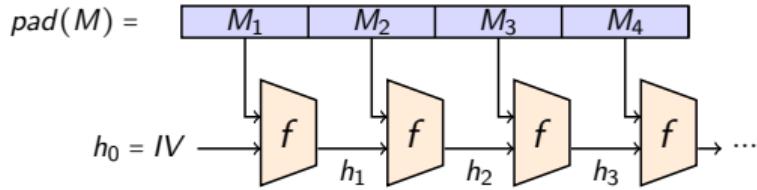
3

- Une collision pour h_2 entraînera une collision sur tous les états suivants
→ Si $md5(m) = md5(m')$, alors $md5(m||a) = md5(m'||a)$
- Dans notre cas :
 - Collision md5 pour m, m' en quelques secondes ⁴
 - 2^{24} candidats a pour avoir les 3 derniers bytes de $md5(m||a)$
 - On à gratuitement la même chose pour $md5(m'||a)$

³[Jean, 2016]

⁴<https://github.com/cr-marcstevens/hashclash>

Construction de Merkle-Damgård et collisions



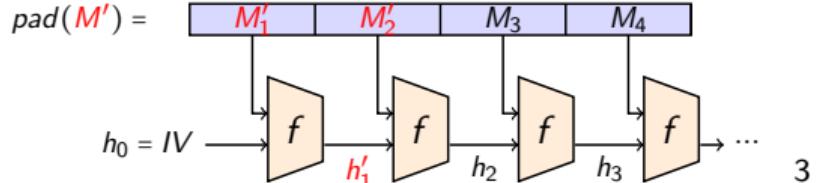
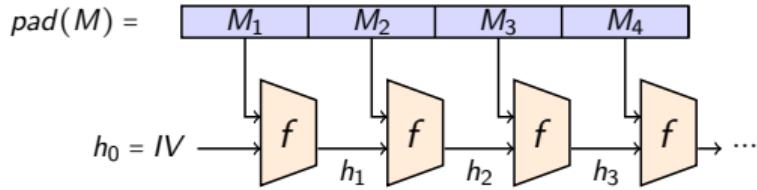
3

- Une collision pour h_2 entraînera une collision sur tous les états suivants
→ Si $md5(m) = md5(m')$, alors $md5(m||a) = md5(m'||a)$
- Dans notre cas :
 - Collision md5 pour m, m' en quelques secondes ⁴
 - 2^{24} candidats a pour avoir les 3 derniers bytes de $md5(m||a)$
 - On à gratuitement la même chose pour $md5(m'||a)$
 - Ceci double les chances d'avoir une collision sur sha1

³[Jean, 2016]

⁴<https://github.com/cr-marcstevens/hashclash>

Construction de Merkle-Damgård et collisions



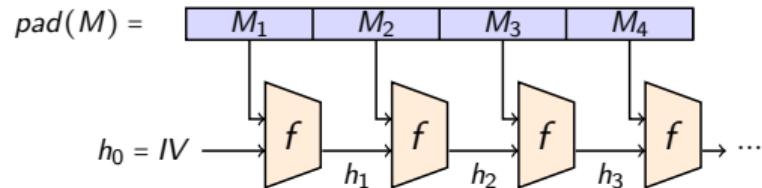
3

- Une collision pour h_2 entraînera une collision sur tous les états suivants
→ Si $md5(m) = md5(m')$, alors $md5(m||a) = md5(m'||a)$
- Dans notre cas :
 - Collision md5 pour m, m' en quelques secondes ⁴
 - 2^{24} candidats a pour avoir les 3 derniers bytes de $md5(m||a)$
 - On à gratuitement la même chose pour $md5(m'||a)$
 - Ceci double les chances d'avoir une collision sur sha1
 - Pourquoi s'arrêter là?

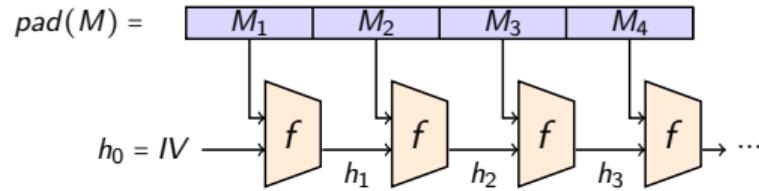
³[Jean, 2016]

⁴<https://github.com/cr-marcstevens/hashclash>

Collisions multiples et md5

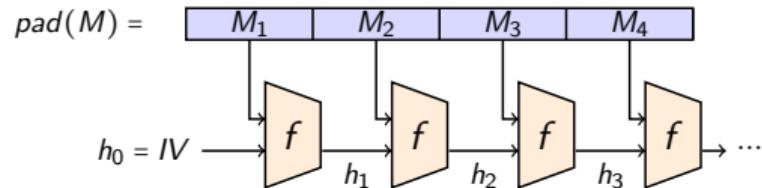


Collisions multiples et md5



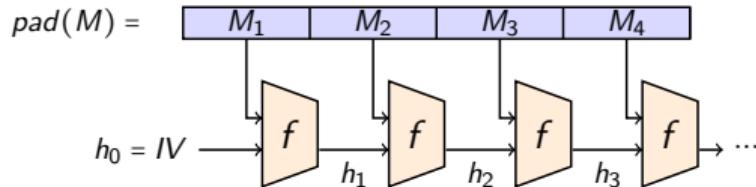
- Pour IV fixé, trouver m_1 et m'_1 en collision sur h_1

Collisions multiples et md5



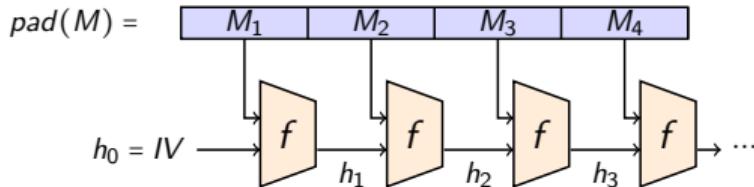
- Pour IV fixé, trouver m_1 et m'_1 en collision sur h_1
- Pour h_1 trouver m_2 et m'_2 en collision sur h_2

Collisions multiples et md5

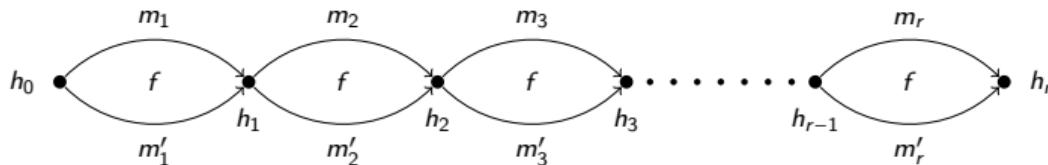


- Pour IV fixé, trouver m_1 et m'_1 en collision sur h_1
- Pour h_1 trouver m_2 et m'_2 en collision sur h_2
- etc

Collisions multiples et md5

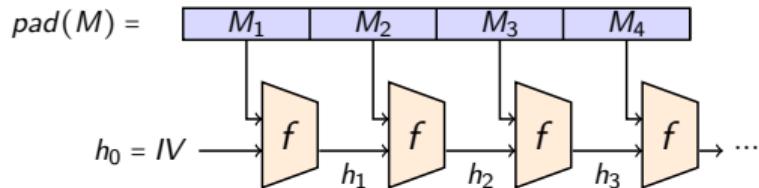


- Pour IV fixé, trouver m_1 et m'_1 en collision sur h_1
- Pour h_1 trouver m_2 et m'_2 en collision sur h_2
- etc

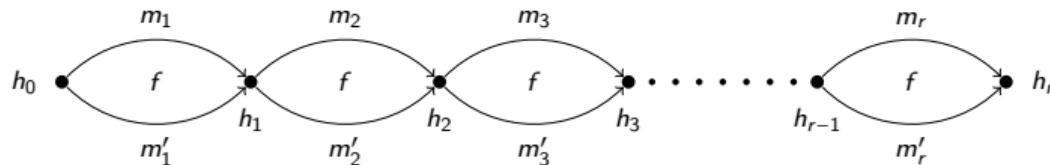


- 2^r collisions sur h_r générées pour le prix de r

Collisions multiples et md5

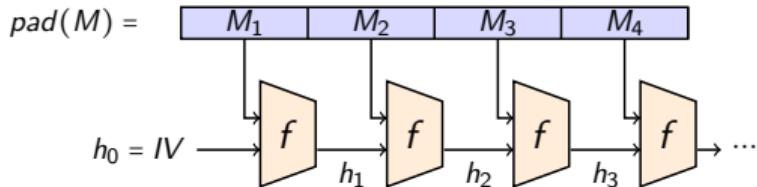


- Pour IV fixé, trouver m_1 et m'_1 en collision sur h_1
- Pour h_1 trouver m_2 et m'_2 en collision sur h_2
- etc

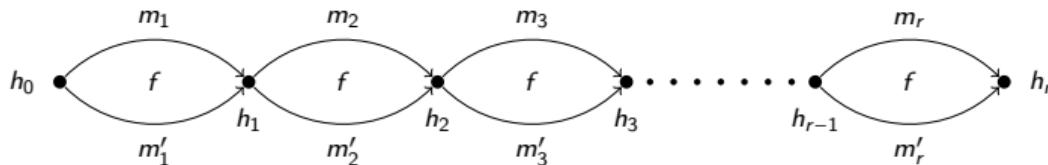


- 2^r collisions sur h_r générées pour le prix de r
- Dans notre cas
 - On génère facilement 2^{24} collisions $md5(x_1) = \dots = md5(x_{2^{24}})$

Collisions multiples et md5

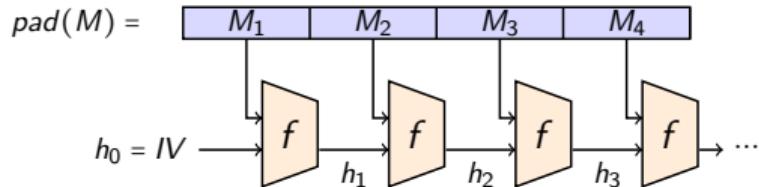


- Pour IV fixé, trouver m_1 et m'_1 en collision sur h_1
- Pour h_1 trouver m_2 et m'_2 en collision sur h_2
- etc

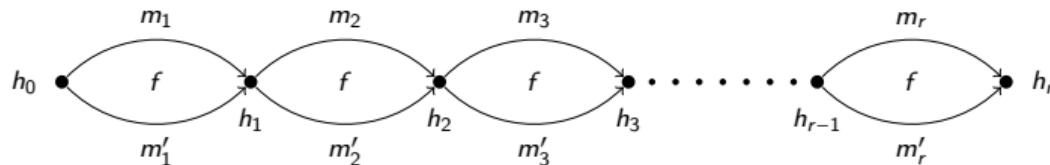


- 2^r collisions sur h_r générées pour le prix de r
- Dans notre cas
 - On génère facilement 2^{24} collisions $md5(x_1) = \dots = md5(x_{2^{24}})$
 - On itère sur 2^{24} valeurs de a jusqu'à avoir la collision sur $md5(x_1||a)$

Collisions multiples et md5

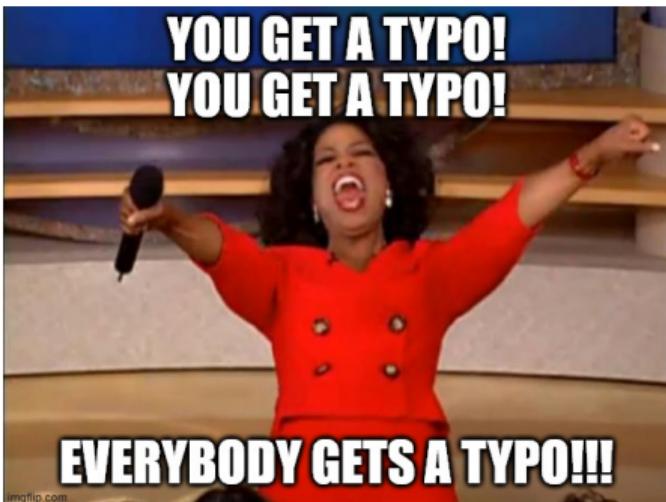


- Pour IV fixé, trouver m_1 et m'_1 en collision sur h_1
- Pour h_1 trouver m_2 et m'_2 en collision sur h_2
- etc



- 2^r collisions sur h_r générées pour le prix de r
- Dans notre cas
 - On génère facilement 2^{24} collisions $md5(x_1) = \dots = md5(x_{2^{24}})$
 - On itère sur 2^{24} valeurs de a jusqu'à avoir la collision sur $md5(x_1||a)$
 - Statistiquement, il y a une collision pour au moins un des $sha1(x_i||a)$

Kyber et les anneaux quotients



- Objectifs :
 - Récupérer k sur 128 bits chiffré via Kyber
- Données :
 - La clef publique A et t
 - Le chiffré $E(k)$
 - Le code source

Kyber / ML-KEM (**le vrai**)

PQ-KEM (chiffrement asymétrique) standardisé par le NIST

Kyber / ML-KEM (le vrai)

PQ-KEM (chiffrement asymétrique) standardisé par le NIST

Learning with Errors (LwE)

Soit $A \in \mathbb{Z}_q^{k \times k}$.

- Facile de récupérer $s \in \mathbb{Z}_q^k$ sachant $t = A.s$ (Interpolation)

Kyber / ML-KEM (le vrai)

PQ-KEM (chiffrement asymétrique) standardisé par le NIST

Ring/Module Learning with Errors (R/MLwE)

Soit $A \in \mathbb{Z}_q^{k \times k}$.

- Facile de récupérer $s \in \mathbb{Z}_q^k$ sachant $t = A.s$ (Interpolation)
- Difficile de récupérer $s \in \mathbb{Z}_q^k$ sachant $t = A.s + e$ où $e \in \mathbb{Z}_q^k$ est petit ($|e| < \eta$)

Kyber / ML-KEM (le vrai)

PQ-KEM (chiffrement asymétrique) standardisé par le NIST

Ring/Module Learning with Errors (R/MLwE)

Soit $A \in \mathbb{Z}_q^{k \times k}$.

- Facile de récupérer $s \in \mathbb{Z}_q^k$ sachant $t = A.s$ (Interpolation)
- Difficile de récupérer $s \in \mathbb{Z}_q^k$ sachant $t = A.s + e$ où $e \in \mathbb{Z}_q^k$ est petit ($|e| < \eta$)
- Même chose dans $\mathcal{R}_q^k = (\mathbb{Z}_q[X]/\langle \phi \rangle)^k$, et ϕ de degré n

Kyber / ML-KEM (le vrai)

PQ-KEM (chiffrement asymétrique) standardisé par le NIST

Ring/Module Learning with Errors (R/MLwE)

Soit $A \in \mathbb{Z}_q^{k \times k}$.

- Facile de récupérer $s \in \mathbb{Z}_q^k$ sachant $t = A.s$ (Interpolation)
- Difficile de récupérer $s \in \mathbb{Z}_q^k$ sachant $t = A.s + e$ où $e \in \mathbb{Z}_q^k$ est petit ($|e| < \eta$)
- Même chose dans $\mathcal{R}_q^k = (\mathbb{Z}_q[X]/\langle \phi \rangle)^k$, et ϕ de degré n

Nom	n	k	q	η
Kyber512	256	2	3329	2
Kyber768	256	3	3329	2
Kyber1024	256	4	3329	2

Kyber / ML-KEM (le vrai)

PQ-KEM (chiffrement asymétrique) standardisé par le NIST

Ring/Module Learning with Errors (R/MLwE)

Soit $A \in \mathbb{Z}_q^{k \times k}$.

- Facile de récupérer $s \in \mathbb{Z}_q^k$ sachant $t = A.s$ (Interpolation)
- Difficile de récupérer $s \in \mathbb{Z}_q^k$ sachant $t = A.s + e$ où $e \in \mathbb{Z}_q^k$ est petit ($|e| < \eta$)
- Même chose dans $\mathcal{R}_q^k = (\mathbb{Z}_q[X]/\langle \phi \rangle)^k$, et ϕ de degré n

Nom	n	k	q	η
Kyber512	256	2	3329	2
Kyber768	256	3	3329	2
Kyber1024	256	4	3329	2

$$\mathcal{R}_q = \mathbb{Z}_q[X]/(X^n + 1)$$

Algorithmes de Kyber

- **Keygen** : $t = A.s + e$
 - Clef privée (s)
 - Clef publique (A, t)

Algorithmes de Kyber

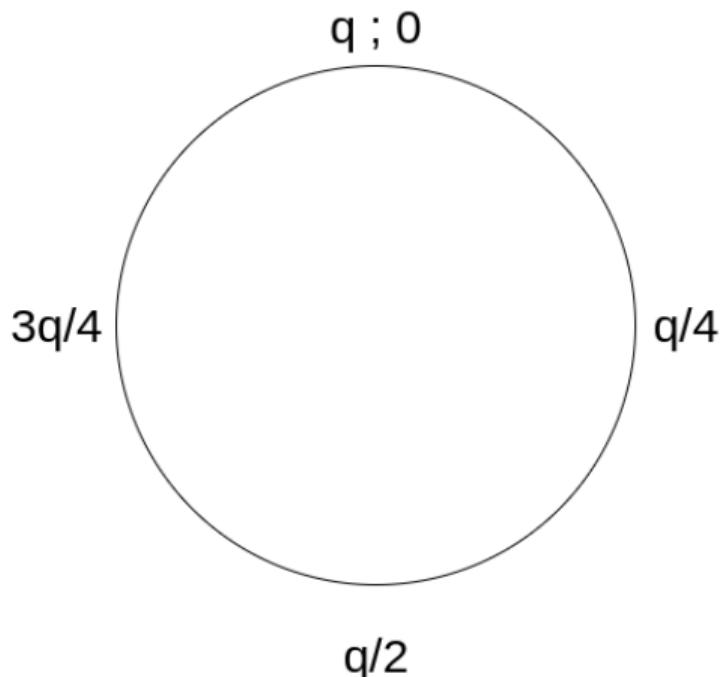
- **Keygen** : $t = A.s + e$
 - Clef privée (s)
 - Clef publique (A, t)
- **Chiffrement** m :
 - $\begin{cases} u &= A^T r + e_1 \\ v &= t^T r + e_2 + m \end{cases}$

Algorithmes de Kyber

- **Keygen** : $t = A.s + e$
 - Clef privée (s)
 - Clef publique (A, t)
- **Chiffrement** m :
 - $\begin{cases} u &= A^T r + e_1 \\ v &= t^T r + e_2 + m \end{cases}$
- **Déchiffrement** (u, v) :
 - $v - s^T u = m + e^T r + e_2 + s^T e_1 = m + e_3$

Algorithmes de Kyber

- **Keygen** : $t = A \cdot s + e$
 - Clef privée (s)
 - Clef publique (A, t)
- **Chiffrement** m :
 - $\begin{cases} u &= A^T r + e_1 \\ v &= t^T r + e_2 + m \end{cases}$
- **Déchiffrement** (u, v) :
 - $v - s^T u = m + e^T r + e_2 + s^T e_1 = m + e_3$

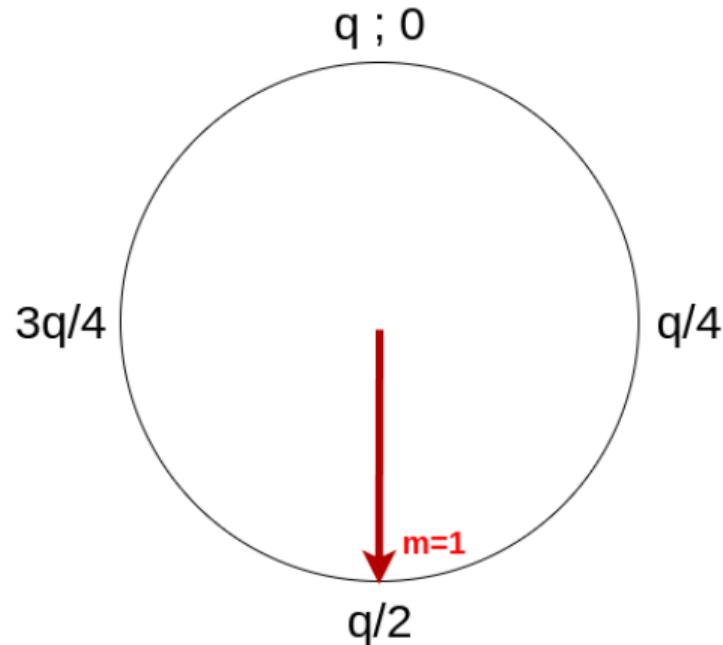


m est un polynôme de \mathcal{R}_q (e.g 0101 devient $0 + \frac{q}{2}x + 0x^2 + \frac{q}{2}x^3$)

Algorithmes de Kyber

- **Keygen** : $t = A.s + e$
 - Clef privée (s)
 - Clef publique (A, t)
- **Chiffrement** m :
 - $\begin{cases} u &= A^T r + e_1 \\ v &= t^T r + e_2 + m \end{cases}$
- **Déchiffrement** (u, v) :
 - $v - s^T u = m + e^T r + e_2 + s^T e_1 = m + e_3$

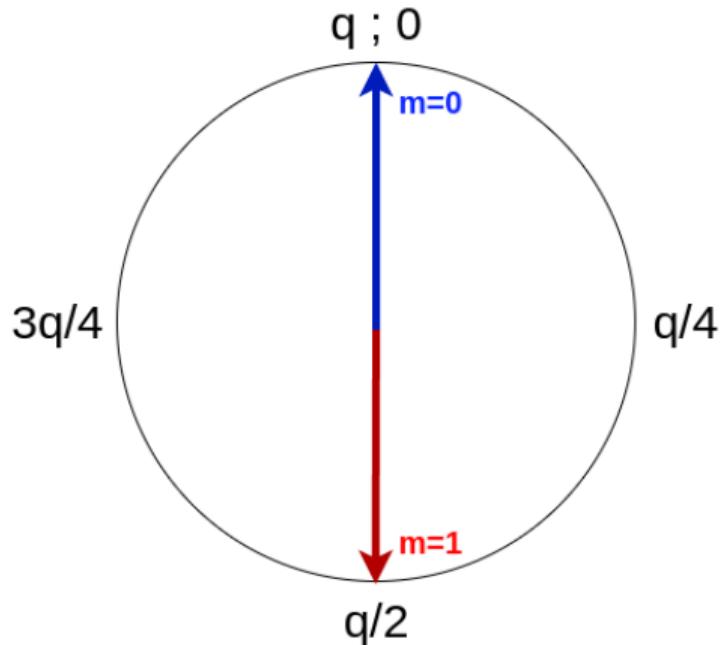
m est un polynôme de \mathcal{R}_q (e.g 0101 devient $0 + \frac{q}{2}x + 0x^2 + \frac{q}{2}x^3$)



Algorithmes de Kyber

- **Keygen** : $t = A.s + e$
 - Clef privée (s)
 - Clef publique (A, t)
- **Chiffrement** m :
 - $\begin{cases} u &= A^T r + e_1 \\ v &= t^T r + e_2 + m \end{cases}$
- **Déchiffrement** (u, v) :
 - $v - s^T u = m + e^T r + e_2 + s^T e_1 = m + e_3$

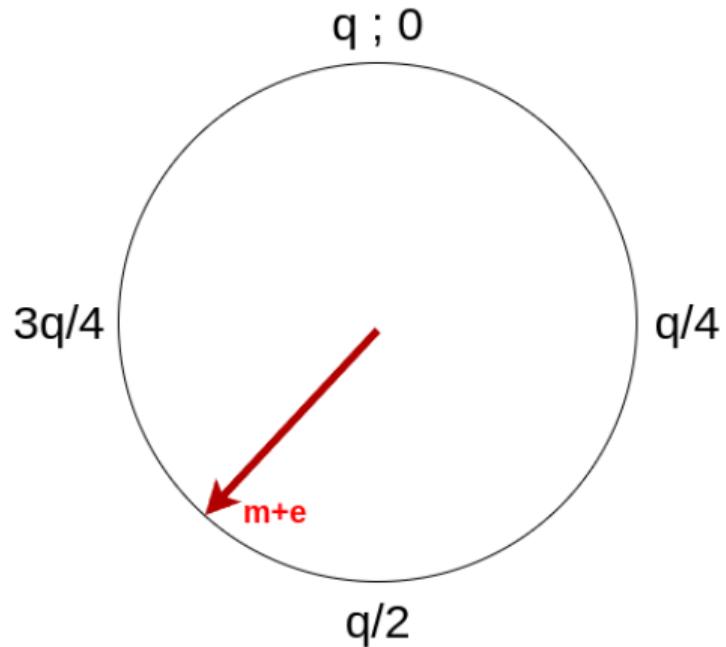
m est un polynôme de \mathcal{R}_q (e.g 0101 devient $0 + \frac{q}{2}x + 0x^2 + \frac{q}{2}x^3$)



Algorithmes de Kyber

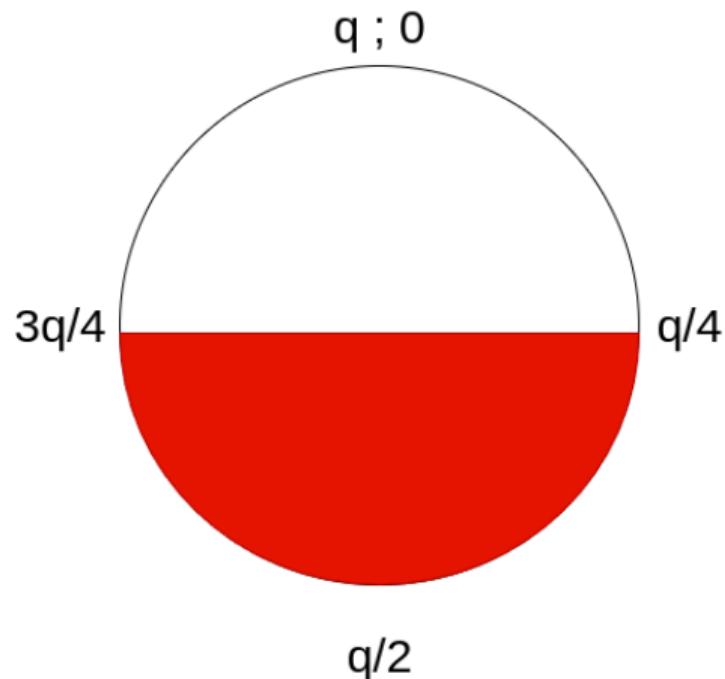
- **Keygen** : $t = A.s + e$
 - Clef privée (s)
 - Clef publique (A, t)
- **Chiffrement** m :
 - $\begin{cases} u &= A^T r + e_1 \\ v &= t^T r + e_2 + m \end{cases}$
- **Déchiffrement** (u, v) :
 - $v - s^T u = m + e^T r + e_2 + s^T e_1 = m + e_3$

m est un polynôme de \mathcal{R}_q (e.g 0101 devient $0 + \frac{q}{2}x + 0x^2 + \frac{q}{2}x^3$)



Algorithmes de Kyber

- **Keygen** : $t = A \cdot s + e$
 - Clef privée (s)
 - Clef publique (A, t)
- **Chiffrement** m :
 - $\begin{cases} u &= A^T r + e_1 \\ v &= t^T r + e_2 + m \end{cases}$
- **Déchiffrement** (u, v) :
 - $v - s^T u = m + e^T r + e_2 + s^T e_1 = m + e_3$



m est un polynôme de \mathcal{R}_q (e.g 0101 devient $0 + \frac{q}{2}x + 0x^2 + \frac{q}{2}x^3$)

Retour au challenge : La (les?) typos

- $E(m_0), \dots, E(m_k)$ au lieu de $E(m_0, \dots, m_k)$
 - m est le polynôme constant 0 ou $q/2$

Retour au challenge : La (les?) typos

- $E(m_0), \dots, E(m_k)$ au lieu de $E(m_0, \dots, m_k)$
 - m est le polynôme constant 0 ou $q/2$
- Travail modulo $X^n - 1$ (au lieu de $X^n + 1$) qui admet des racines dans \mathbb{Z}_q

Retour au challenge : La (les?) typos

- $E(m_0), \dots, E(m_k)$ au lieu de $E(m_0, \dots, m_k)$
 - m est le polynôme constant 0 ou $q/2$
- Travail modulo $X^n - 1$ (au lieu de $X^n + 1$) qui admet des racines dans \mathbb{Z}_q
 - $P \mapsto P(1)$ est un morphisme d'anneau sur \mathcal{R}_q
 - C'est le morphisme $P \mapsto P \bmod X - 1$

Retour au challenge : La (les?) typos

- $E(m_0), \dots, E(m_k)$ au lieu de $E(m_0, \dots, m_k)$
 - m est le polynôme constant 0 ou $q/2$
- Travail modulo $X^n - 1$ (au lieu de $X^n + 1$) qui admet des racines dans \mathbb{Z}_q
 - $P \mapsto P(1)$ est un morphisme d'anneau sur \mathcal{R}_q
 - C'est le morphisme $P \mapsto P \bmod X - 1$
 - $v(1) = [\textcolor{blue}{t}_0(1), \textcolor{blue}{t}_1(1)] \bullet [\textcolor{red}{r}_0(1), \textcolor{red}{r}_1(1)]^T + \textcolor{red}{e}_2(1) + m(1)$

Retour au challenge : La (les?) typos

- $E(m_0), \dots, E(m_k)$ au lieu de $E(m_0, \dots, m_k)$
 - m est le polynôme constant 0 ou $q/2$
- Travail modulo $X^n - 1$ (au lieu de $X^n + 1$) qui admet des racines dans \mathbb{Z}_q
 - $P \mapsto P(1)$ est un morphisme d'anneau sur \mathcal{R}_q
 - C'est le morphisme $P \mapsto P \bmod X - 1$
 - $v(1) = [\textcolor{blue}{t}_0(1), \textcolor{blue}{t}_1(1)] \bullet [\textcolor{red}{r}_0(1), \textcolor{red}{r}_1(1)]^T + \textcolor{red}{e}_2(1) + m(1) \in [-q/2, q/2]$

Retour au challenge : La (les?) typos

- $E(m_0), \dots, E(m_k)$ au lieu de $E(m_0, \dots, m_k)$
 - m est le polynôme constant 0 ou $q/2$
- Travail modulo $X^n - 1$ (au lieu de $X^n + 1$) qui admet des racines dans \mathbb{Z}_q
 - $P \mapsto P(1)$ est un morphisme d'anneau sur \mathcal{R}_q
 - C'est le morphisme $P \mapsto P \bmod X - 1$
 - $v(1) = [\textcolor{blue}{t}_0(1), \textcolor{blue}{t}_1(1)] \bullet [\textcolor{red}{r}_0(1), \textcolor{red}{r}_1(1)]^T + \textcolor{red}{e}_2(1) + m(1) \in [-q/2, q/2]$
 - $v(1) = 0 + \textcolor{red}{e}_2(1) + m$ avec probabilité $1/q^2$ $(\sim \frac{1}{11\,082\,241})$

Retour au challenge : La (les?) typos

- $E(m_0), \dots, E(m_k)$ au lieu de $E(m_0, \dots, m_k)$
 - m est le polynôme constant 0 ou $q/2$
 - Travail modulo $X^n - 1$ (au lieu de $X^n + 1$) qui admet des racines dans \mathbb{Z}_q
 - $P \mapsto P(1)$ est un morphisme d'anneau sur \mathcal{R}_q
 - C'est le morphisme $P \mapsto P \bmod X - 1$
 - $v(1) = [\textcolor{blue}{t_0}(1), \textcolor{blue}{t_1}(1)] \bullet [\textcolor{red}{r_0}(1), \textcolor{red}{r_1}(1)]^T + \textcolor{red}{e_2}(1) + m(1) \in [-q/2, q/2]$
 - $v(1) = 0 + \textcolor{red}{e_2}(1) + m$ avec probabilité $1/q^2$
 - $v(1) = \epsilon \textcolor{red}{r_0}(1) + \epsilon \textcolor{red}{r_1}(1) + \textcolor{red}{e_2}(1) + m$ avec probabilité ϵ^2/q^2
- ($\sim \frac{1}{11\,082\,241}$)
($\sim \frac{1}{12\,313}$)

Retour au challenge : La (les?) typos

- $E(m_0), \dots, E(m_k)$ au lieu de $E(m_0, \dots, m_k)$
 - m est le polynôme constant 0 ou $q/2$
 - Travail modulo $X^n - 1$ (au lieu de $X^n + 1$) qui admet des racines dans \mathbb{Z}_q
 - $P \mapsto P(1)$ est un morphisme d'anneau sur \mathcal{R}_q
 - C'est le morphisme $P \mapsto P \bmod X - 1$
 - $v(1) = [\textcolor{blue}{t_0}(1), \textcolor{blue}{t_1}(1)] \bullet [\textcolor{red}{r_0}(1), \textcolor{red}{r_1}(1)]^T + \textcolor{red}{e_2}(1) + m(1) \in [-q/2, q/2]$
 - $v(1) = 0 + \textcolor{red}{e_2}(1) + m$ avec probabilité $1/q^2$
 - $v(1) = \epsilon \textcolor{red}{r_0}(1) + \epsilon \textcolor{red}{r_1}(1) + \textcolor{red}{e_2}(1) + m$ avec probabilité ϵ^2/q^2
- ($\sim \frac{1}{11\,082\,241}$)
($\sim \frac{1}{12\,313}$)

Retour au challenge : La (les?) typos

- $E(m_0), \dots, E(m_k)$ au lieu de $E(m_0, \dots, m_k)$
 - m est le polynôme constant 0 ou $q/2$
- Travail modulo $X^n - 1$ (au lieu de $X^n + 1$) qui admet des racines dans \mathbb{Z}_q
 - $P \mapsto P(1)$ est un morphisme d'anneau sur \mathcal{R}_q
 - C'est le morphisme $P \mapsto P \pmod{X - 1}$
 - $v(1) = [\textcolor{blue}{t}_0(1), \textcolor{blue}{t}_1(1)] \bullet [\textcolor{red}{r}_0(1), \textcolor{red}{r}_1(1)]^T + \textcolor{red}{e}_2(1) + m(1) \in [-q/2, q/2]$
 - $v(1) = 0 + \textcolor{red}{e}_2(1) + m$ avec probabilité $1/q^2$
 - $v(1) = \epsilon \textcolor{red}{r}_0(1) + \epsilon \textcolor{red}{r}_1(1) + \textcolor{red}{e}_2(1) + m$ avec probabilité ϵ^2/q^2
 - $P \mapsto P(-1)$ est aussi un morphisme d'anneau
 - C'est le morphisme $P \mapsto P \pmod{X + 1}$

$$(\sim \frac{1}{11\,082\,241})$$
$$(\sim \frac{1}{12\,313})$$

$$(\sim \frac{1}{6\,156})$$

Retour au challenge : La (les?) typos

- $E(m_0), \dots, E(m_k)$ au lieu de $E(m_0, \dots, m_k)$
 - m est le polynôme constant 0 ou $q/2$
- Travail modulo $X^n - 1$ (au lieu de $X^n + 1$) qui admet des racines dans \mathbb{Z}_q
 - $P \mapsto P(1)$ est un morphisme d'anneau sur \mathcal{R}_q
 - C'est le morphisme $P \mapsto P \pmod{X - 1}$
 - $v(1) = [\textcolor{blue}{t}_0(1), \textcolor{blue}{t}_1(1)] \bullet [\textcolor{red}{r}_0(1), \textcolor{red}{r}_1(1)]^T + \textcolor{red}{e}_2(1) + m(1) \in [-q/2, q/2]$
 - $v(1) = 0 + \textcolor{red}{e}_2(1) + m$ avec probabilité $1/q^2$
 - $v(1) = \epsilon \textcolor{red}{r}_0(1) + \epsilon \textcolor{red}{r}_1(1) + \textcolor{red}{e}_2(1) + m$ avec probabilité ϵ^2/q^2
 - $P \mapsto P(-1)$ est aussi un morphisme d'anneau
 - C'est le morphisme $P \mapsto P \pmod{X + 1}$
 - $X^{256} - 1$ à exactement 256 racines dans \mathbb{Z}_q (et $X^{256} + 1$ n'en a pas)
 - Difficile à exploiter car $\textcolor{red}{e}_2(r)$ est grand

$$(\sim \frac{1}{11\,082\,241})$$
$$(\sim \frac{1}{12\,313})$$

$$(\sim \frac{1}{6\,156})$$

Retour au challenge : La (les?) typos

- $E(m_0), \dots, E(m_k)$ au lieu de $E(m_0, \dots, m_k)$
 - m est le polynôme constant 0 ou $q/2$
- Travail modulo $X^n - 1$ (au lieu de $X^n + 1$) qui admet des racines dans \mathbb{Z}_q
 - $P \mapsto P(1)$ est un morphisme d'anneau sur \mathcal{R}_q
 - C'est le morphisme $P \mapsto P \pmod{X - 1}$
 - $v(1) = [\textcolor{blue}{t}_0(1), \textcolor{blue}{t}_1(1)] \bullet [\textcolor{red}{r}_0(1), \textcolor{red}{r}_1(1)]^T + \textcolor{red}{e}_2(1) + m(1) \in [-q/2, q/2]$
 - $v(1) = 0 + \textcolor{red}{e}_2(1) + m$ avec probabilité $1/q^2$
 - $v(1) = \epsilon r_0(1) + \epsilon r_1(1) + \textcolor{red}{e}_2(1) + m$ avec probabilité ϵ^2/q^2
 - $P \mapsto P(-1)$ est aussi un morphisme d'anneau
 - C'est le morphisme $P \mapsto P \pmod{X + 1}$
 - $X^{256} - 1$ à exactement 256 racines dans \mathbb{Z}_q (et $X^{256} + 1$ n'en a pas)
 - Difficile à exploiter car $\textcolor{red}{e}_2(r)$ est grand
 - $X^{256} - 1$ est divisible par $X^{2^k} - 1$ (et $X^{256} + 1$ n'en a pas)

$$(\sim \frac{1}{11\,082\,241})$$
$$(\sim \frac{1}{12\,313})$$

$$(\sim \frac{1}{6\,156})$$

$$(\sim \frac{1}{96})$$

Retour au challenge : La (les?) typos

- $E(m_0), \dots, E(m_k)$ au lieu de $E(m_0, \dots, m_k)$
 - m est le polynôme constant 0 ou $q/2$
- Travail modulo $X^n - 1$ (au lieu de $X^n + 1$) qui admet des racines dans \mathbb{Z}_q
 - $P \mapsto P(1)$ est un morphisme d'anneau sur \mathcal{R}_q
 - C'est le morphisme $P \mapsto P \pmod{X - 1}$
 - $v(1) = [\textcolor{blue}{t}_0(1), \textcolor{blue}{t}_1(1)] \bullet [\textcolor{red}{r}_0(1), \textcolor{red}{r}_1(1)]^T + \textcolor{red}{e}_2(1) + m(1) \in [-q/2, q/2]$
 - $v(1) = 0 + \textcolor{red}{e}_2(1) + m$ avec probabilité $1/q^2$ $(\sim \frac{1}{11\,082\,241})$
 - $v(1) = \epsilon \textcolor{red}{r}_0(1) + \epsilon \textcolor{red}{r}_1(1) + \textcolor{red}{e}_2(1) + m$ avec probabilité ϵ^2/q^2 $(\sim \frac{1}{12\,313})$
 - $P \mapsto P(-1)$ est aussi un morphisme d'anneau $(\sim \frac{1}{6\,156})$
 - C'est le morphisme $P \mapsto P \pmod{X + 1}$
 - $X^{256} - 1$ à exactement 256 racines dans \mathbb{Z}_q (et $X^{256} + 1$ n'en a pas)
 - Difficile à exploiter car $\textcolor{red}{e}_2(r)$ est grand
 - $X^{256} - 1$ est divisible par $X^{2^k} - 1$ (et $X^{256} + 1$ n'en a pas) $(\sim \frac{1}{96})$
- On peut générer des instances du problème jusqu'à avoir la solution via au moins l'un de ces morphismes 

Analyse différentielle et boomerang

Attaque sur SbPN



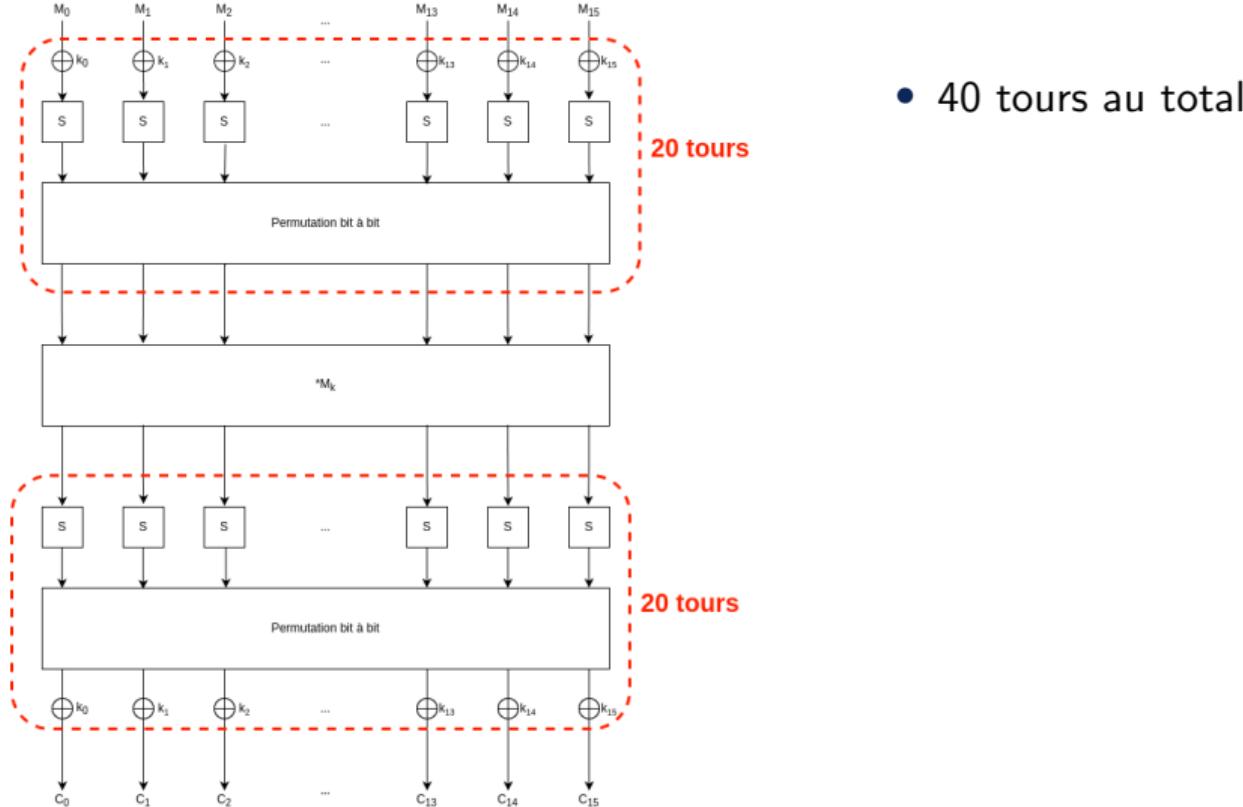
- Objectifs :
 - Forger un couple clair / chiffré sans connaître la clef



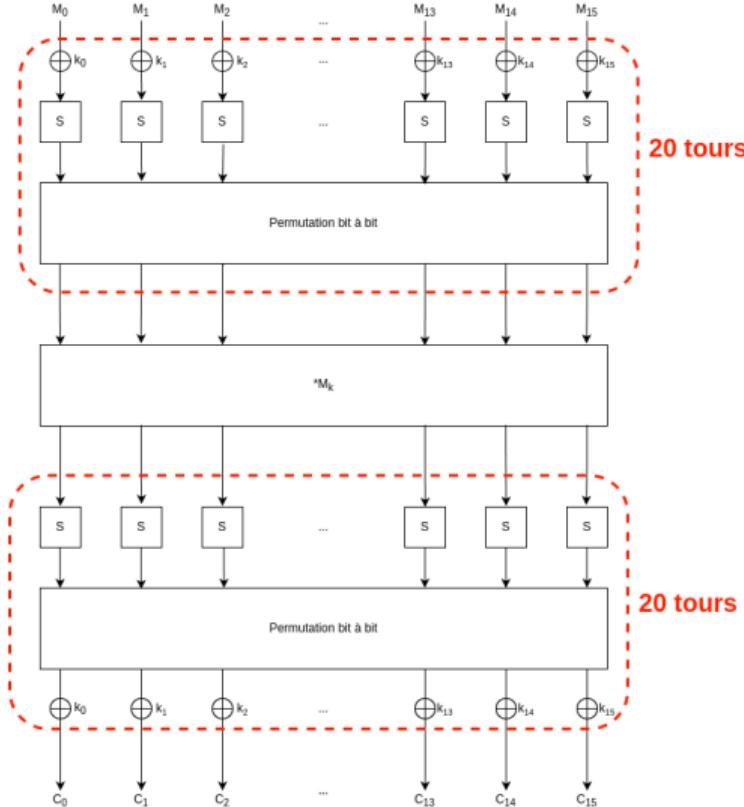
WHEN I ROLL MY OWN EVIL SYMMETRIC CRYPTO

- Objectifs :
 - Forger un couple clair / chiffré sans connaître la clef
- Données
 - Exactement 3 oracles de chiffrement / déchiffrement
 - Le code source

Le cipher Jafar

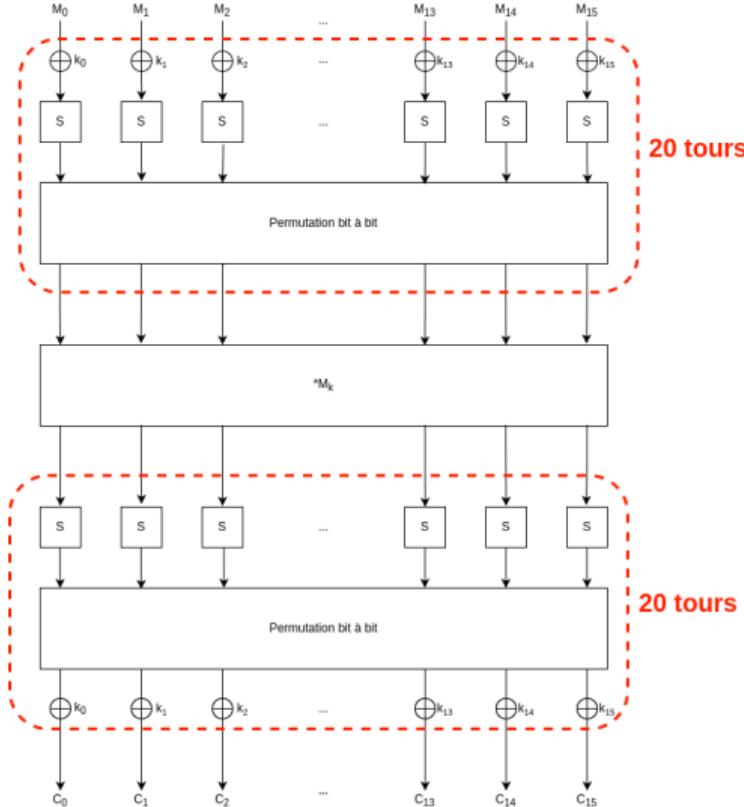


Le cipher Jafar



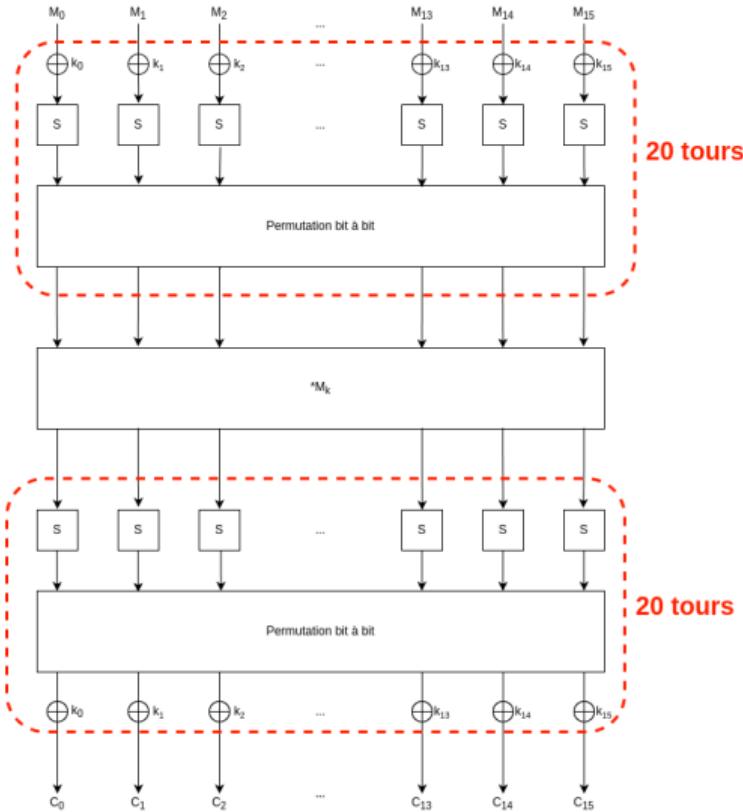
- 40 tours au total
- Pas de tour de clef

Le cipher Jafar



- 40 tours au total
- Pas de tour de clef
- Tout est linéaire à part la SBox

Le cipher Jafar



- 40 tours au total
- Pas de tour de clef
- Tout est linéaire à part la SBox

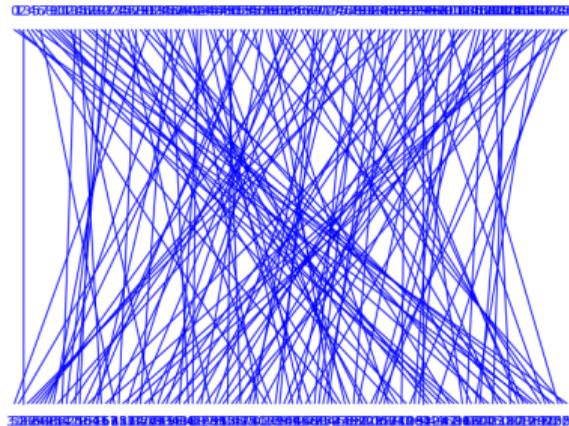


@bluesheet

Fausse piste : la permutation

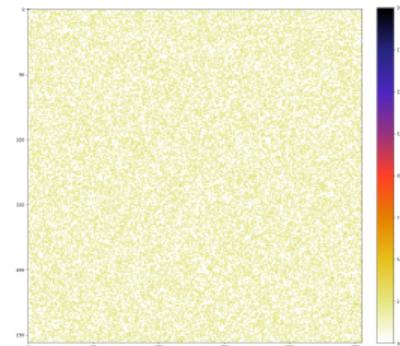
```
P_Perm = Permutation(P)
P_perm.show("braid")
P_perm.fixed_points()
P_perm.cycle_tuples()
```

- Deux points fixes (2 et 94)
- Trois cycles non triviaux de taille
82, 31, 13



Analyse de la SBox

```
from sage.crypto.sbox import SBox  
sb = SBox(S)  
sb.difference_distribution_table()
```

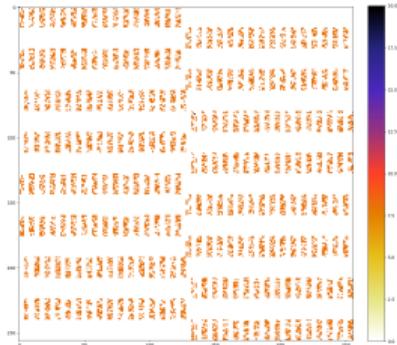
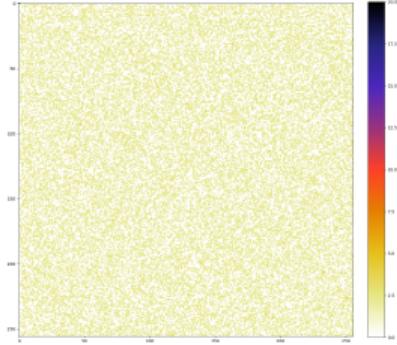


- DDT[a,b] := Probabilité que $S[x \oplus a] = S[x] \oplus b$

Analyse de la SBox

```
from sage.crypto.sbox import SBox  
sb = SBox(S)  
sb.difference_distribution_table()
```

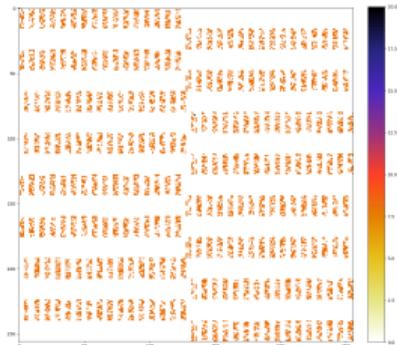
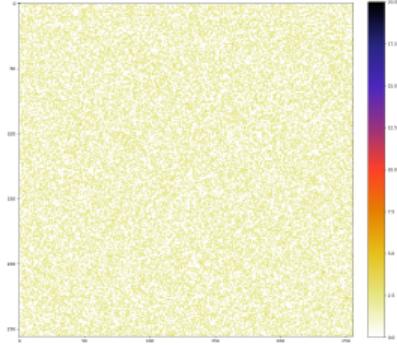
- DDT[a,b] := Probabilité que $S[x \oplus a] = S[x] \oplus b$
→ 4 différentielles par Sbox: $(0 \mapsto 0)$, $(24 \mapsto 129)$, $(74 \mapsto 7)$,
 $(82 \mapsto 134)$



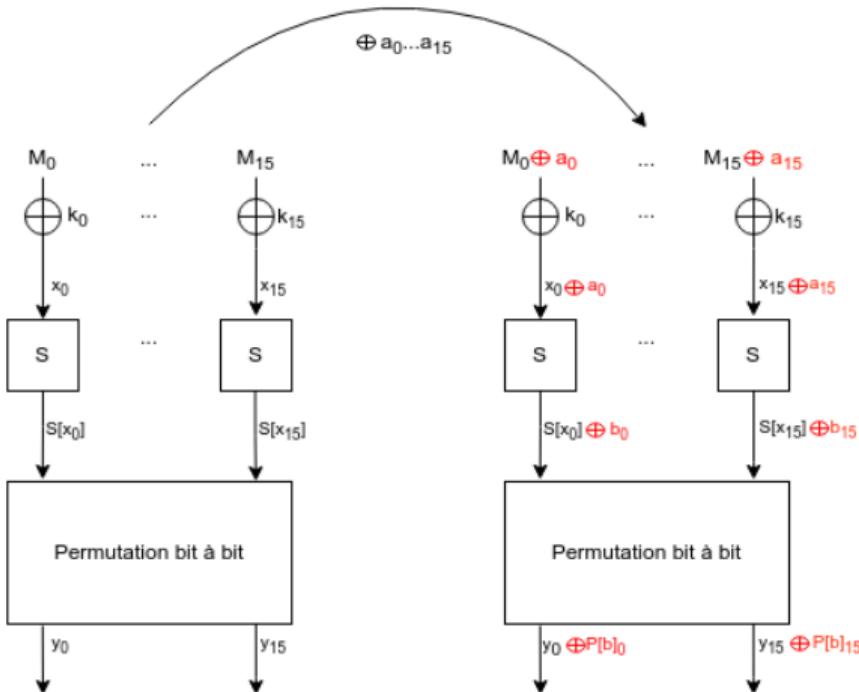
Analyse de la SBox

```
from sage.crypto.sbox import SBox  
sb = SBox(S)  
sb.difference_distribution_table()
```

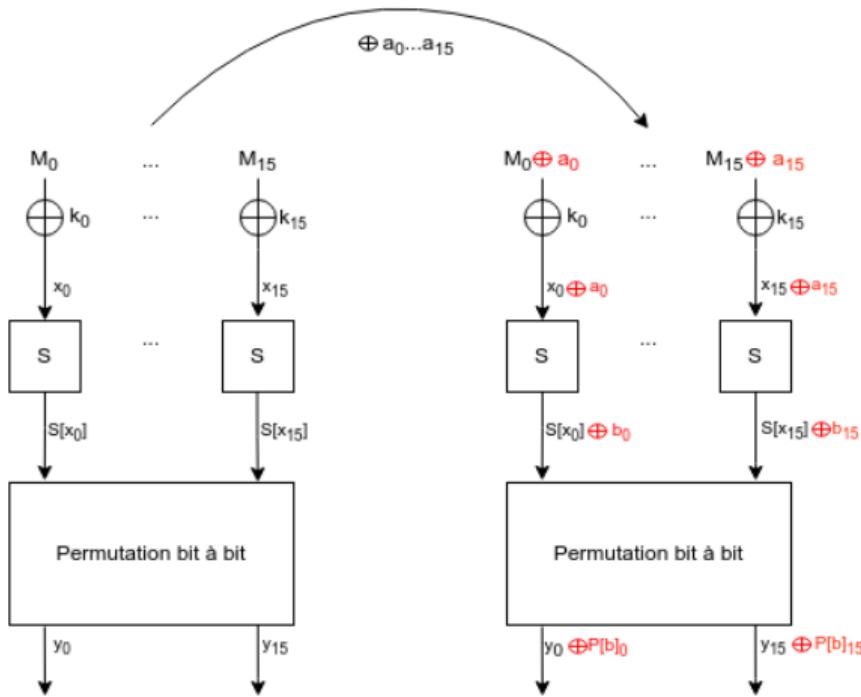
- DDT[a,b] := Probabilité que $S[x \oplus a] = S[x] \oplus b$
→ 4 différentielles par Sbox: $(0 \mapsto 0)$, $(24 \mapsto 129)$, $(74 \mapsto 7)$,
 $(82 \mapsto 134)$
- 16 octets $\Rightarrow 2^{32}$ différentielles à tester



Introduire une différentielle

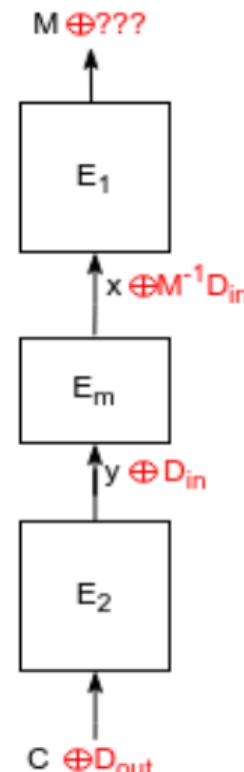
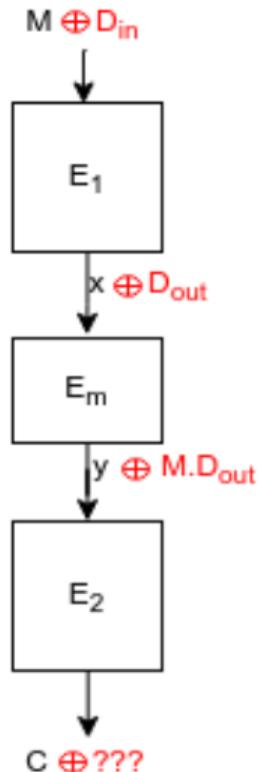
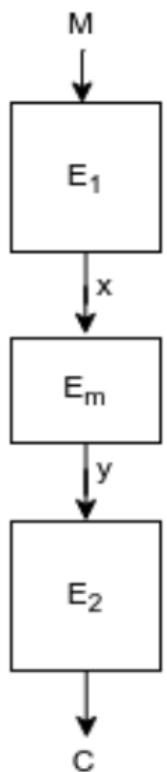


Introduire une différentielle

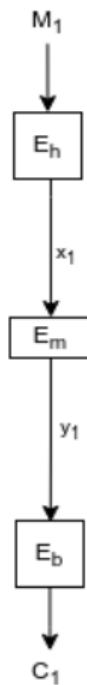


- $D_{in} = [0, 82, 0, 24, 0, 0, 0, 82, 0, 0, 0, 0, 0, 0, 0, 0]$
- $D_{out} = [0, 0, 0, 0, 0, 74, 0, 0, 0, 24, 0, 0, 74, 0, 0]$

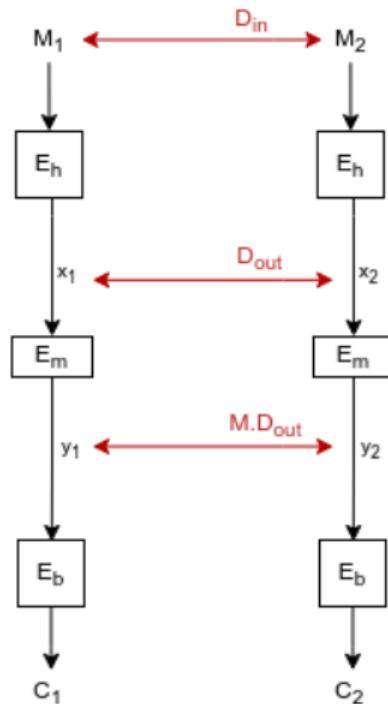
Sur 20 rounds



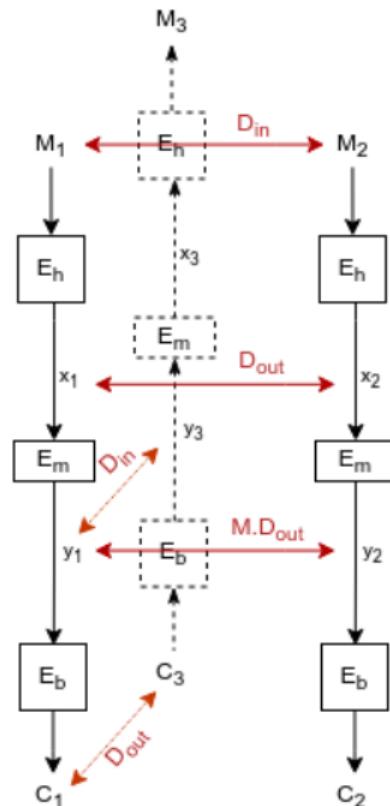
Introduction au boomerang



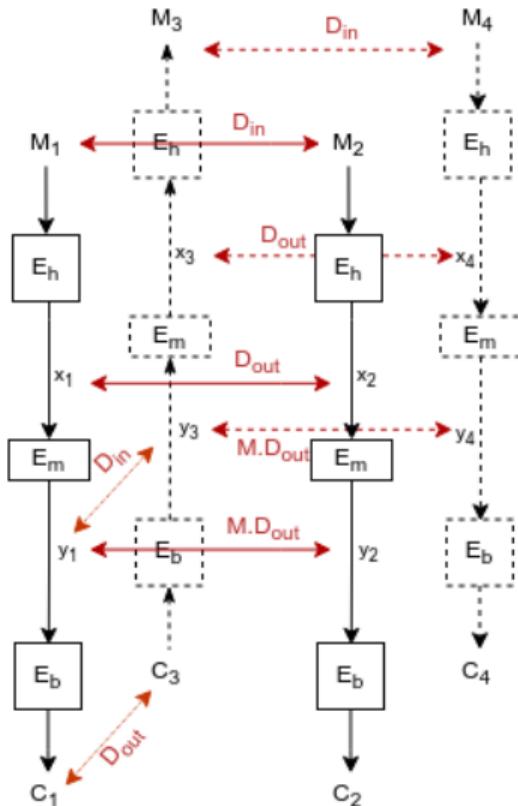
Introduction au boomerang



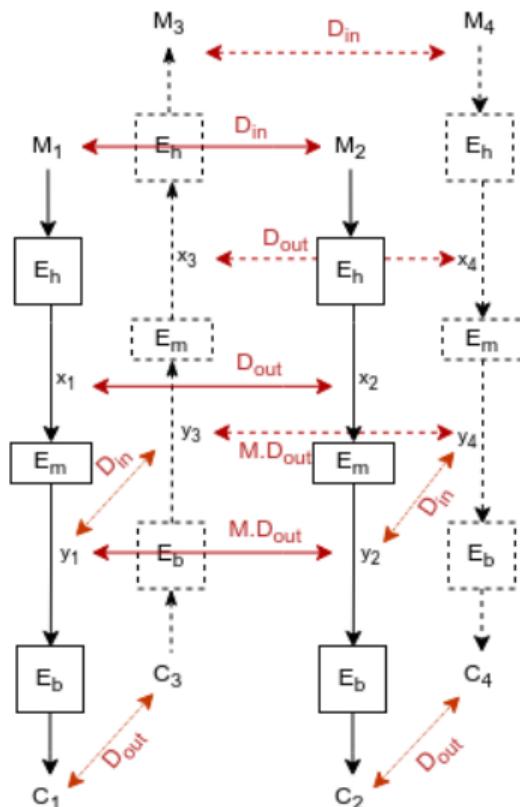
Introduction au boomerang



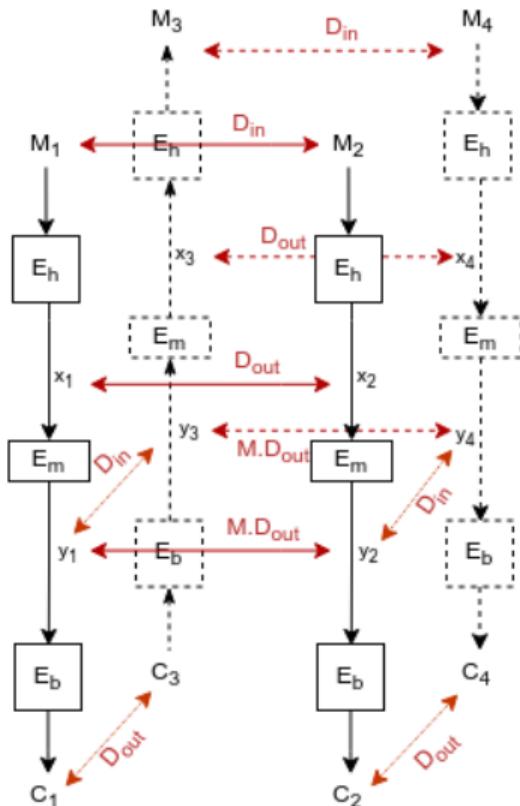
Introduction au boomerang



Introduction au boomerang



Introduction au boomerang



Le schéma multivarié UOV

Comment faire une (mauvaise) vinaigrette





- Objectifs :
 - Forger une signature valide
 - Sans la clef privée



- Objectifs :
 - Forger une signature valide
 - Sans la clef privée
 - Sans la clef publique



- Objectifs :
 - Forger une signature valide
 - Sans la clef privée
 - **Sans la clef publique**
- Données :
 - 1600 signatures

Sur les représentations des fonctions quadratiques

Fonction quadratique

$$f(x_1, \dots, x_n) = \sum_{i \leq j} a_{ij} x_i x_j + \sum_i b_i x_i + c$$

Sur les représentations des fonctions quadratiques

Fonction quadratique

Forme quadratique

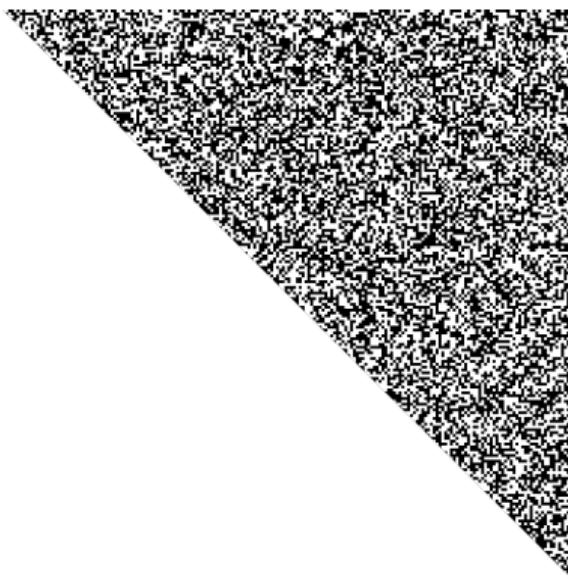
$$f'(\mathbf{x}, \mathbf{x}) = \sum_{i \leq j} a_{ij} x_i x_j$$

Sur les représentations des fonctions quadratiques

Fonction quadratique

Forme quadratique

$$f'(\mathbf{x}, \mathbf{x}) = \sum_{i \leq j} a_{ij} x_i x_j = \mathbf{x}^T \mathbf{A} \mathbf{x}$$



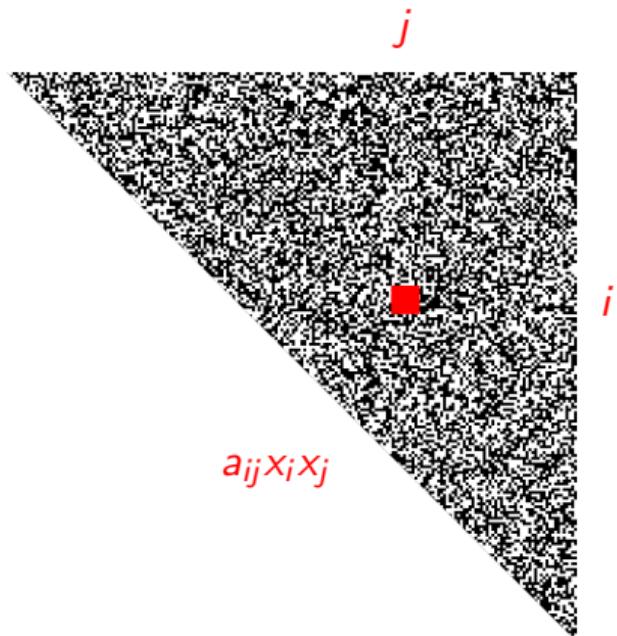
Représentation matricielle d'une forme quadratique

Sur les représentations des fonctions quadratiques

Fonction quadratique

Forme quadratique

$$f'(\mathbf{x}, \mathbf{x}) = \sum_{i \leq j} a_{ij} x_i x_j = \mathbf{x}^T \mathbf{A} \mathbf{x}$$



Représentation matricielle d'une forme quadratique

Sur les formes bilinéaires

$$f(x_1, \dots, x_n) = \sum_{i \leq j} a_{ij} x_i x_j + \sum_i b_i x_i + c$$

Forme bilinéaire

$$f'(\mathbf{x}, \mathbf{y}) = \sum_{i \leq j} a_{ij} x_i y_j$$

Sur les formes bilinéaires

$$f(x_1, \dots, x_n) = \sum_{i \leq j} a_{ij} x_i x_j + \sum_i b_i x_i + c$$

Forme bilinéaire

$$f'(\mathbf{x}, \mathbf{y}) = \sum_{i \leq j} a_{ij} x_i y_j = \mathbf{x}^T \mathbf{G} \mathbf{y}$$

Sur les formes bilinéaires

$$f(x_1, \dots, x_n) = \sum_{i \leq j} a_{ij} x_i x_j + \sum_i b_i x_i + c$$

Forme bilinéaire

$$f'(\mathbf{x}, \mathbf{y}) = \sum_{i \leq j} a_{ij} x_i y_j = \mathbf{x}^T \mathbf{G} \mathbf{y}$$

/!\ G est de taille maximale

Sur les formes bilinéaires

$$f(x_1, \dots, x_n) = \sum_{i \leq j} a_{ij} x_i x_j + \sum_i b_i x_i + c$$

Forme bilinéaire

$$f'(\mathbf{x}, \mathbf{y}) = \sum_{i \leq j} a_{ij} x_i y_j = \mathbf{x}^T \mathbf{G} \mathbf{y}$$

/!\ G est de taille maximale

- Pour \mathbf{y} fixé, une équation en \mathbf{x} est linéaire (et vice versa)

Sur les formes bilinéaires

$$f(x_1, \dots, x_n) = \sum_{i \leq j} a_{ij} x_i x_j + \sum_i b_i x_i + c$$

Forme bilinéaire

$$f'(\mathbf{x}, \mathbf{y}) = \sum_{i \leq j} a_{ij} x_i y_j = \mathbf{x}^T \mathbf{G} \mathbf{y}$$

/!\ G est de taille maximale

- Pour \mathbf{y} fixé, une équation en \mathbf{x} est linéaire (et vice versa)

Composition

$$f'(\mathbf{x}, \mathbf{y}) = f(\mathbf{x} + \mathbf{y}) - f(\mathbf{x}) - f(\mathbf{y}) + f(\mathbf{0})$$
 est une forme bilinéaire symétrique

- Les parties linéaires et constantes se simplifient
- $(\mathbf{x} + \mathbf{y})^T \mathbf{A} (\mathbf{x} + \mathbf{y}) = \mathbf{x}^T \mathbf{A} \mathbf{x} + \mathbf{y}^T \mathbf{A} \mathbf{y} + \mathbf{x}^T (\mathbf{A} + \mathbf{A}^T) \mathbf{y}$

Une signature via UOV (Unbalanced Oil and Vinegar)

La clef publique UOV

$\mathcal{Q} = (\mathcal{Q}_1, \dots, \mathcal{Q}_h)$ des fonctions quadratiques à n variables

$$\mathcal{Q}_i(\mathbf{y}) = \mathbf{y}^T \mathbf{A}_i \mathbf{y} + \mathbf{b}_i^T \mathbf{y} + c_i$$

Une signature via UOV (Unbalanced Oil and Vinegar)

La clef publique UOV

$\mathcal{Q} = (\mathcal{Q}_1, \dots, \mathcal{Q}_h)$ des fonctions quadratiques à n variables

$$\mathcal{Q}_i(\mathbf{y}) = \mathbf{y}^T \mathbf{A}_i \mathbf{y} + \mathbf{b}_i^T \mathbf{y} + c_i$$

La signature de $\mathbf{t} = (t_1, \dots, t_h)$ est une solution $\mathbf{y} = (y_1, \dots, y_n)$ du système $\mathcal{Q}(\mathbf{y}) = \mathbf{t}$:

$$\begin{cases} \mathcal{Q}_1(\mathbf{y}) = \mathcal{Q}_1(y_1, \dots, y_n) = t_1 \\ \vdots \\ \mathcal{Q}_h(\mathbf{y}) = \mathcal{Q}_h(y_1, \dots, y_n) = t_h \end{cases}$$

Une signature via UOV (Unbalanced Oil and Vinegar)

La clef publique UOV

$\mathcal{Q} = (\mathcal{Q}_1, \dots, \mathcal{Q}_h)$ des fonctions quadratiques à n variables

$$\mathcal{Q}_i(\mathbf{y}) = \mathbf{y}^T \mathbf{A}_i \mathbf{y} + \mathbf{b}_i^T \mathbf{y} + c_i$$

La signature de $\mathbf{t} = (t_1, \dots, t_h)$ est une solution $\mathbf{y} = (y_1, \dots, y_n)$ du système $\mathcal{Q}(\mathbf{y}) = \mathbf{t}$:

$$\begin{cases} \mathcal{Q}_1(\mathbf{y}) = \mathcal{Q}_1(y_1, \dots, y_n) = t_1 \\ \vdots \\ \mathcal{Q}_h(\mathbf{y}) = \mathcal{Q}_h(y_1, \dots, y_n) = t_h \end{cases}$$

Difficulté du problème UOV

Pour des bonnes valeurs de h , v et $n = h + v$, il est difficile de résoudre un tel système quadratique quelconque.

Une signature via UOV (Unbalanced Oil and Vinegar)

La clef publique UOV

$\mathcal{Q} = (\mathcal{Q}_1, \dots, \mathcal{Q}_h)$ des fonctions quadratiques à n variables

$$\mathcal{Q}_i(\mathbf{y}) = \mathbf{y}^T \mathbf{A}_i \mathbf{y} + \mathbf{b}_i^T \mathbf{y} + c_i$$

La signature de $\mathbf{t} = (t_1, \dots, t_h)$ est une solution $\mathbf{y} = (y_1, \dots, y_n)$ du système $\mathcal{Q}(\mathbf{y}) = \mathbf{t}$:

$$\begin{cases} \mathcal{Q}_1(\mathbf{y}) = \mathcal{Q}_1(y_1, \dots, y_n) = t_1 \\ \vdots \\ \mathcal{Q}_h(\mathbf{y}) = \mathcal{Q}_h(y_1, \dots, y_n) = t_h \end{cases}$$

Difficulté du problème UOV

Pour des bonnes valeurs de h , v et $n = h + v$, il est difficile de résoudre un tel système quadratique quelconque.

Sans clef privée, il est difficile de signer un message

La trapdoor d'UOV

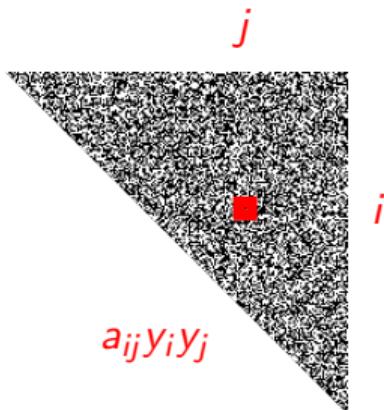
$$\mathbb{F}_q^n = \mathbb{F}_q^h \oplus \mathbb{F}_q^\nu = \mathcal{H} \oplus \mathcal{V} \text{ (huile + vinaigre)}$$

La trapdoor d'UOV

$$\mathbb{F}_q^n = \mathbb{F}_q^h \oplus \mathbb{F}_q^\nu = \mathcal{H} \oplus \mathcal{V} \text{ (huile + vinaigre)} \Rightarrow \mathbf{x} = (x_1, \dots, x_h, x_{h+1}, \dots, x_{h+\nu}) = (\mathbf{x}_{\mathcal{H}}, \mathbf{x}_{\mathcal{V}})$$

La trapdoor d'UOV

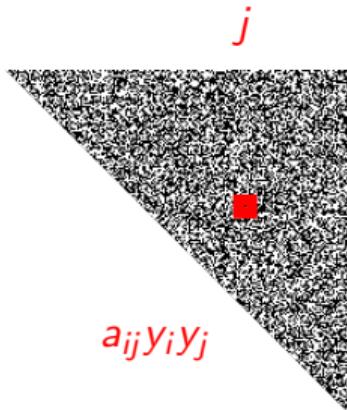
$$\mathbb{F}_q^n = \mathbb{F}_q^h \oplus \mathbb{F}_q^\nu = \mathcal{H} \oplus \mathcal{V} \text{ (huile + vinaigre)} \Rightarrow \mathbf{x} = (x_1, \dots, x_h, x_{h+1}, \dots, x_{h+\nu}) = (\mathbf{x}_{\mathcal{H}}, \mathbf{x}_{\mathcal{V}})$$



Partie quadratique de Q_i publique

La trapdoor d'UOV

$$\mathbb{F}_q^n = \mathbb{F}_q^h \oplus \mathbb{F}_q^v = \mathcal{H} \oplus \mathcal{V} \text{ (huile + vinaigre)} \Rightarrow \mathbf{x} = (x_1, \dots, x_h, x_{h+1}, \dots, x_{h+v}) = (\mathbf{x}_{\mathcal{H}}, \mathbf{x}_{\mathcal{V}})$$



Partie quadratique de Q_i publique

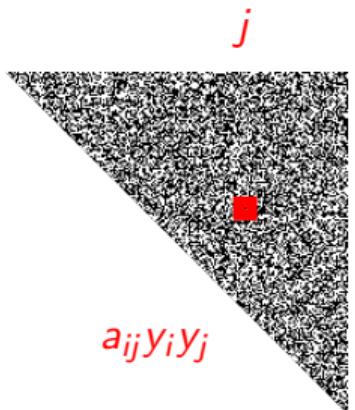
$$\mathbf{y} = \mathcal{S}(\mathbf{x})$$
$$\Leftarrow$$



Partie quadratique de $\mathcal{P}_i = \mathcal{Q}_i \circ \mathcal{S}$ privée

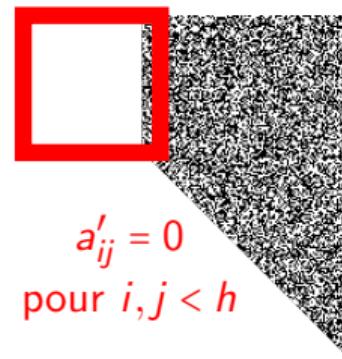
La trapdoor d'UOV

$$\mathbb{F}_q^n = \mathbb{F}_q^h \oplus \mathbb{F}_q^\nu = \mathcal{H} \oplus \mathcal{V} \text{ (huile + vinaigre)} \Rightarrow \mathbf{x} = (x_1, \dots, x_h, x_{h+1}, \dots, x_{h+\nu}) = (\mathbf{x}_{\mathcal{H}}, \mathbf{x}_{\mathcal{V}})$$



Partie quadratique de \mathcal{Q}_i publique

$$\mathbf{y} = \mathcal{S}(\mathbf{x})$$



$$a'_{ij} = 0$$

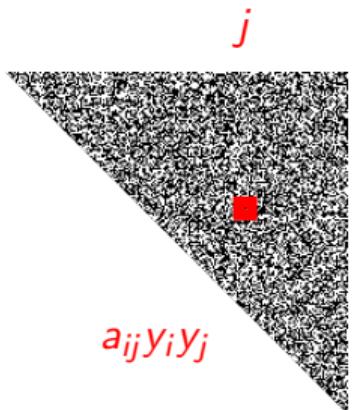
pour $i, j < h$

Partie quadratique de $\mathcal{P}_i = \mathcal{Q}_i \circ \mathcal{S}$ privée

- Par construction, $\mathcal{P}_i(\mathbf{x}) = \mathcal{P}_i((\mathbf{x}_{\mathcal{H}}, \mathbf{x}_{\mathcal{V}}))$ est linéaire en $\mathbf{x}_{\mathcal{H}}$ pour $\mathbf{x}_{\mathcal{V}}$ fixé.

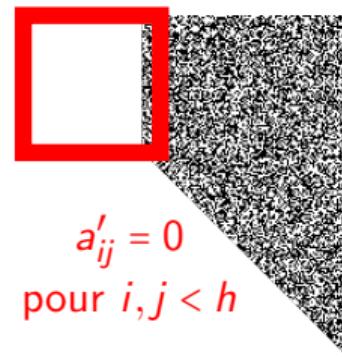
La trapdoor d'UOV

$$\mathbb{F}_q^n = \mathbb{F}_q^h \oplus \mathbb{F}_q^\nu = \mathcal{H} \oplus \mathcal{V} \text{ (huile + vinaigre)} \Rightarrow \mathbf{x} = (x_1, \dots, x_h, x_{h+1}, \dots, x_{h+\nu}) = (\mathbf{x}_{\mathcal{H}}, \mathbf{x}_{\mathcal{V}})$$



Partie quadratique de \mathcal{Q}_i publique

$$\mathbf{y} = \mathcal{S}(\mathbf{x})$$

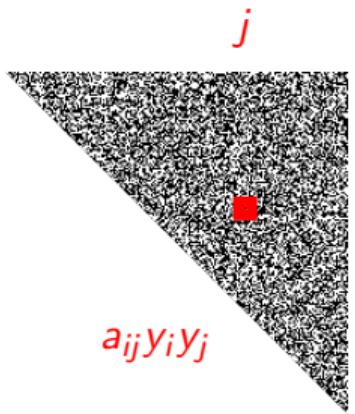


Partie quadratique de $\mathcal{P}_i = \mathcal{Q}_i \circ \mathcal{S}$ privée

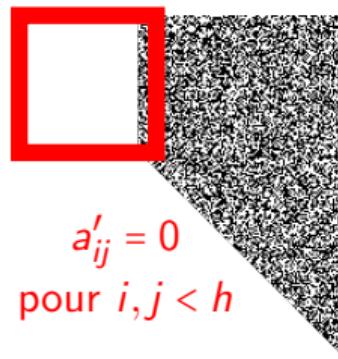
- Par construction, $\mathcal{P}_i(\mathbf{x}) = \mathcal{P}_i((\mathbf{x}_{\mathcal{H}}, \mathbf{x}_{\mathcal{V}}))$ est linéaire en $\mathbf{x}_{\mathcal{H}}$ pour $\mathbf{x}_{\mathcal{V}}$ fixé.
- Pour $\mathbf{x}_{\mathcal{V}}$ fixé, il est trivial de résoudre $\mathcal{P}(\mathbf{x}_{\mathcal{H}}, \mathbf{x}_{\mathcal{V}}) = \mathbf{t}$

La trapdoor d'UOV

$$\mathbb{F}_q^n = \mathbb{F}_q^h \oplus \mathbb{F}_q^\nu = \mathcal{H} \oplus \mathcal{V} \text{ (huile + vinaigre)} \Rightarrow \mathbf{x} = (x_1, \dots, x_h, x_{h+1}, \dots, x_{h+\nu}) = (\mathbf{x}_{\mathcal{H}}, \mathbf{x}_{\mathcal{V}})$$



$$\mathbf{y} = \mathcal{S}(\mathbf{x})$$



$$a'_{ij} = 0$$

pour $i, j < h$

Partie quadratique de $\mathcal{P}_i = \mathcal{Q}_i \circ \mathcal{S}$ privée

- Par construction, $\mathcal{P}_i(\mathbf{x}) = \mathcal{P}_i((\mathbf{x}_{\mathcal{H}}, \mathbf{x}_{\mathcal{V}}))$ est linéaire en $\mathbf{x}_{\mathcal{H}}$ pour $\mathbf{x}_{\mathcal{V}}$ fixé.
- Pour $\mathbf{x}_{\mathcal{V}}$ fixé, il est trivial de résoudre $\mathcal{P}(\mathbf{x}_{\mathcal{H}}, \mathbf{x}_{\mathcal{V}}) = \mathbf{t}$
- Pour $\mathbf{y} = \mathcal{S}(\mathbf{x})$ sera une solution valide de $\mathcal{Q}(\mathbf{y}) = \mathbf{t}$

Récapitulatif d'UOV

Récapitulatif d'UOV

Keygen

- Générer un changement de variable linéaire S aléatoire (matrice $n \times n$)

Récapitulatif d'UOV

Keygen

- Générer un changement de variable linéaire \mathcal{S} aléatoire (matrice $n \times n$)
- Générer $\mathcal{P}_{i < h}$ quadratique tels que $\mathcal{P}_i((\mathbf{x}_{\mathcal{H}}, \mathbf{x}_{\mathcal{V}}))$ est linéaire en $\mathbf{x}_{\mathcal{H}}$ pour $\mathbf{x}_{\mathcal{V}}$ fixé

Récapitulatif d'UOV

Keygen

- Générer un changement de variable linéaire \mathcal{S} aléatoire (matrice $n \times n$)
- Générer $\mathcal{P}_{i < h}$ quadratique tels que $\mathcal{P}_i((\mathbf{x}_{\mathcal{H}}, \mathbf{x}_{\mathcal{V}}))$ est linéaire en $\mathbf{x}_{\mathcal{H}}$ pour $\mathbf{x}_{\mathcal{V}}$ fixé
- **Clef publique** : $\mathcal{Q} = \mathcal{P} \circ \mathcal{S}^{-1}$ avec $h\binom{n+1}{2}$ coefficients

Récapitulatif d'UOV

Keygen

- Générer un changement de variable linéaire \mathcal{S} aléatoire (matrice $n \times n$)
- Générer $\mathcal{P}_{i < h}$ quadratique tels que $\mathcal{P}_i((\mathbf{x}_{\mathcal{H}}, \mathbf{x}_{\mathcal{V}}))$ est linéaire en $\mathbf{x}_{\mathcal{H}}$ pour $\mathbf{x}_{\mathcal{V}}$ fixé
- **Clef publique** : $\mathcal{Q} = \mathcal{P} \circ \mathcal{S}^{-1}$ avec $h \binom{n+1}{2}$ coefficients
- **Clef privée** : \mathcal{P}, \mathcal{S} avec $h \left(\binom{n+1}{2} - \binom{h}{2} \right)$ et n^2 coefficients

Récapitulatif d'UOV

Keygen

- Générer un changement de variable linéaire \mathcal{S} aléatoire (matrice $n \times n$)
- Générer $\mathcal{P}_{i < h}$ quadratique tels que $\mathcal{P}_i((\mathbf{x}_{\mathcal{H}}, \mathbf{x}_{\mathcal{V}}))$ est linéaire en $\mathbf{x}_{\mathcal{H}}$ pour $\mathbf{x}_{\mathcal{V}}$ fixé
- **Clef publique** : $\mathcal{Q} = \mathcal{P} \circ \mathcal{S}^{-1}$ avec $h \binom{n+1}{2}$ coefficients
- **Clef privée** : \mathcal{P}, \mathcal{S} avec $h \left(\binom{n+1}{2} - \binom{h}{2} \right)$ et n^2 coefficients

Signature de m

Récapitulatif d'UOV

Keygen

- Générer un changement de variable linéaire \mathcal{S} aléatoire (matrice $n \times n$)
- Générer $\mathcal{P}_{i < h}$ quadratique tels que $\mathcal{P}_i((\mathbf{x}_{\mathcal{H}}, \mathbf{x}_{\mathcal{V}}))$ est linéaire en $\mathbf{x}_{\mathcal{H}}$ pour $\mathbf{x}_{\mathcal{V}}$ fixé
- **Clef publique** : $\mathcal{Q} = \mathcal{P} \circ \mathcal{S}^{-1}$ avec $h \binom{n+1}{2}$ coefficients
- **Clef privée** : \mathcal{P}, \mathcal{S} avec $h \left(\binom{n+1}{2} - \binom{h}{2} \right)$ et n^2 coefficients

Signature de m

- Encoder m comme $\mathbf{t} \in \mathbb{F}_q^h$

Récapitulatif d'UOV

Keygen

- Générer un changement de variable linéaire \mathcal{S} aléatoire (matrice $n \times n$)
- Générer $\mathcal{P}_{i < h}$ quadratique tels que $\mathcal{P}_i((\mathbf{x}_{\mathcal{H}}, \mathbf{x}_{\mathcal{V}}))$ est linéaire en $\mathbf{x}_{\mathcal{H}}$ pour $\mathbf{x}_{\mathcal{V}}$ fixé
- **Clef publique** : $\mathcal{Q} = \mathcal{P} \circ \mathcal{S}^{-1}$ avec $h \binom{n+1}{2}$ coefficients
- **Clef privée** : \mathcal{P}, \mathcal{S} avec $h \left(\binom{n+1}{2} - \binom{h}{2} \right)$ et n^2 coefficients

Signature de m

- Encoder m comme $\mathbf{t} \in \mathbb{F}_q^h$
- Fixer aléatoirement $\mathbf{x}_{\mathcal{V}} \in \mathcal{V}$

Récapitulatif d'UOV

Keygen

- Générer un changement de variable linéaire \mathcal{S} aléatoire (matrice $n \times n$)
- Générer $\mathcal{P}_{i < h}$ quadratique tels que $\mathcal{P}_i((\mathbf{x}_{\mathcal{H}}, \mathbf{x}_{\mathcal{V}}))$ est linéaire en $\mathbf{x}_{\mathcal{H}}$ pour $\mathbf{x}_{\mathcal{V}}$ fixé
- **Clef publique** : $\mathcal{Q} = \mathcal{P} \circ \mathcal{S}^{-1}$ avec $h \binom{n+1}{2}$ coefficients
- **Clef privée** : \mathcal{P}, \mathcal{S} avec $h \left(\binom{n+1}{2} - \binom{h}{2} \right)$ et n^2 coefficients

Signature de m

- Encoder m comme $\mathbf{t} \in \mathbb{F}_q^h$
- Fixer aléatoirement $\mathbf{x}_{\mathcal{V}} \in \mathcal{V}$
- Résoudre $\mathcal{P}((\mathbf{x}_{\mathcal{H}}, \mathbf{x}_{\mathcal{V}})) = \mathbf{t}$ (linéaire en $\mathbf{x}_{\mathcal{H}}$). Si pas de solution, choisir un autre $\mathbf{x}_{\mathcal{V}}$

Récapitulatif d'UOV

Keygen

- Générer un changement de variable linéaire \mathcal{S} aléatoire (matrice $n \times n$)
- Générer $\mathcal{P}_{i < h}$ quadratique tels que $\mathcal{P}_i((\mathbf{x}_{\mathcal{H}}, \mathbf{x}_{\mathcal{V}}))$ est linéaire en $\mathbf{x}_{\mathcal{H}}$ pour $\mathbf{x}_{\mathcal{V}}$ fixé
- **Clef publique** : $\mathcal{Q} = \mathcal{P} \circ \mathcal{S}^{-1}$ avec $h \binom{n+1}{2}$ coefficients
- **Clef privée** : \mathcal{P}, \mathcal{S} avec $h \left(\binom{n+1}{2} - \binom{h}{2} \right)$ et n^2 coefficients

Signature de m

- Encoder m comme $\mathbf{t} \in \mathbb{F}_q^h$
- Fixer aléatoirement $\mathbf{x}_{\mathcal{V}} \in \mathcal{V}$
- Résoudre $\mathcal{P}((\mathbf{x}_{\mathcal{H}}, \mathbf{x}_{\mathcal{V}})) = \mathbf{t}$ (linéaire en $\mathbf{x}_{\mathcal{H}}$). Si pas de solution, choisir un autre $\mathbf{x}_{\mathcal{V}}$
- **Signature**: $\mathbf{y} = \mathcal{S}(\mathbf{x})$ est une solution du système $\mathcal{Q}(\mathbf{y}) = \mathbf{t}$

Retour au challenge

Parametres

- $n = 60$, donc polynômes à 60 variables, vecteurs \mathbf{x} , \mathbf{y} de dimensions 60
- $h = 24$, donc 24 équations quadratiques \mathcal{P}_i et \mathcal{Q}_i
- $v = n - h = 36$, donc 36 variables de \mathbf{x}_V à fixer pour la signature

Retour au challenge

Paramètres

- $n = 60$, donc polynômes à 60 variables, vecteurs \mathbf{x} , \mathbf{y} de dimensions 60
 - $h = 24$, donc 24 équations quadratiques \mathcal{P}_i et \mathcal{Q}_i
 - $v = n - h = 36$, donc 36 variables de \mathbf{x}_V à fixer pour la signature
-
- Analyse du code source

Keygen: La clef privée inverse h et v (au lieu d'avoir 24 inconnues linéaires dans la clef privée, on en a 36)

Retour au challenge

Paramètres

- $n = 60$, donc polynômes à 60 variables, vecteurs \mathbf{x} , \mathbf{y} de dimensions 60
 - $h = 24$, donc 24 équations quadratiques \mathcal{P}_i et \mathcal{Q}_i
 - $v = n - h = 36$, donc 36 variables de \mathbf{x}_v à fixer pour la signature
-
- Analyse du code source

Keygen: La clef privée inverse h et v (au lieu d'avoir 24 inconnues linéaires dans la clef privée, on en a 36)

Signature: les 36 valeurs de \mathbf{x}_v sont déterministes, calculées via SHAKE256 du message, donc connues

Sur la confidentialité des vecteurs vinaigres⁵

⁵[Pébereau, 2023]

Sur la confidentialité des vecteurs vinaigres⁵

$\mathbf{y} = \mathcal{S}(\mathbf{x}) = \mathcal{S}(\mathbf{x}_{\mathcal{V}}) + \mathcal{S}(\mathbf{x}_{\mathcal{H}}) = \mathcal{S}_{|\mathcal{V}}(\mathbf{x}_{\mathcal{V}}) + \mathcal{S}_{|\mathcal{H}}(\mathbf{x}_{\mathcal{H}})$ car $\mathbb{F}_q^n = \mathcal{V} \oplus \mathcal{H}$

Avec $\mathcal{S}_{|\mathcal{V}} : \mathcal{V} \longrightarrow \mathcal{S}(\mathcal{V}) := \mathcal{V}'$ et $\mathcal{S}_{|\mathcal{H}} : \mathcal{H} \longrightarrow \mathcal{S}(\mathcal{H}) := \mathcal{H}'$ des *isomorphismes*

⁵[Pébereau, 2023]

Sur la confidentialité des vecteurs vinaigres⁵

$\mathbf{y} = \mathcal{S}(\mathbf{x}) = \mathcal{S}(\mathbf{x}_{\mathcal{V}}) + \mathcal{S}(\mathbf{x}_{\mathcal{H}}) = \mathcal{S}|_{\mathcal{V}}(\mathbf{x}_{\mathcal{V}}) + \mathcal{S}|_{\mathcal{H}}(\mathbf{x}_{\mathcal{H}})$ car $\mathbb{F}_q^n = \mathcal{V} \oplus \mathcal{H}$

Avec $\mathcal{S}|_{\mathcal{V}} : \mathcal{V} \longrightarrow \mathcal{S}(\mathcal{V}) := \mathcal{V}'$ et $\mathcal{S}|_{\mathcal{H}} : \mathcal{H} \longrightarrow \mathcal{S}(\mathcal{H}) := \mathcal{H}'$ des *isomorphismes*

Ceci donne $\mathcal{S}|_{\mathcal{V}}^*(\mathbf{y}) = \mathbf{x}_{\mathcal{V}} + 0$

⁵[Pébereau, 2023]

Sur la confidentialité des vecteurs vinaigres⁵

$\mathbf{y} = \mathcal{S}(\mathbf{x}) = \mathcal{S}(\mathbf{x}_{\mathcal{V}}) + \mathcal{S}(\mathbf{x}_{\mathcal{H}}) = \mathcal{S}|_{\mathcal{V}}(\mathbf{x}_{\mathcal{V}}) + \mathcal{S}|_{\mathcal{H}}(\mathbf{x}_{\mathcal{H}})$ car $\mathbb{F}_q^n = \mathcal{V} \oplus \mathcal{H}$

Avec $\mathcal{S}|_{\mathcal{V}} : \mathcal{V} \longrightarrow \mathcal{S}(\mathcal{V}) := \mathcal{V}'$ et $\mathcal{S}|_{\mathcal{H}} : \mathcal{H} \longrightarrow \mathcal{S}(\mathcal{H}) := \mathcal{H}'$ des *isomorphismes*

Ceci donne $\mathcal{S}|_{\mathcal{V}}^*(\mathbf{y}) = \mathbf{x}_{\mathcal{V}} + 0 \longrightarrow$ On interpole $\mathcal{S}|_{\mathcal{V}}$ donc \mathcal{V}' avec $36 * 60 / 36 = 60$ signatures

⁵[Pébereau, 2023]

Sur la confidentialité des vecteurs vinaigres⁵

$\mathbf{y} = \mathcal{S}(\mathbf{x}) = \mathcal{S}(\mathbf{x}_{\mathcal{V}}) + \mathcal{S}(\mathbf{x}_{\mathcal{H}}) = \mathcal{S}|_{\mathcal{V}}(\mathbf{x}_{\mathcal{V}}) + \mathcal{S}|_{\mathcal{H}}(\mathbf{x}_{\mathcal{H}})$ car $\mathbb{F}_q^n = \mathcal{V} \oplus \mathcal{H}$

Avec $\mathcal{S}|_{\mathcal{V}} : \mathcal{V} \longrightarrow \mathcal{S}(\mathcal{V}) := \mathcal{V}'$ et $\mathcal{S}|_{\mathcal{H}} : \mathcal{H} \longrightarrow \mathcal{S}(\mathcal{H}) := \mathcal{H}'$ des *isomorphismes*

Ceci donne $\mathcal{S}|_{\mathcal{V}}^*(\mathbf{y}) = \mathbf{x}_{\mathcal{V}} + 0 \longrightarrow$ On interpole $\mathcal{S}|_{\mathcal{V}}$ donc \mathcal{V}' avec $36 * 60 / 36 = 60$ signatures

Comme \mathcal{P} est linéaire sur \mathcal{H} , après changement de variable $\mathcal{Q} = \mathcal{P} \circ \mathcal{S}$ sera linéaire sur \mathcal{H}'

Forger une signature UOV

$\mathbb{F}_q^n = \mathcal{V}' + \mathcal{H}'$, donc connaître l'espace \mathcal{V}' ou \mathcal{H}' suffit à forger une signature UOV

⁵[Pébereau, 2023]

Sur la confidentialité des vecteurs vinaigres⁵

$\mathbf{y} = \mathcal{S}(\mathbf{x}) = \mathcal{S}(\mathbf{x}_{\mathcal{V}}) + \mathcal{S}(\mathbf{x}_{\mathcal{H}}) = \mathcal{S}|_{\mathcal{V}}(\mathbf{x}_{\mathcal{V}}) + \mathcal{S}|_{\mathcal{H}}(\mathbf{x}_{\mathcal{H}})$ car $\mathbb{F}_q^n = \mathcal{V} \oplus \mathcal{H}$

Avec $\mathcal{S}|_{\mathcal{V}} : \mathcal{V} \longrightarrow \mathcal{S}(\mathcal{V}) := \mathcal{V}'$ et $\mathcal{S}|_{\mathcal{H}} : \mathcal{H} \longrightarrow \mathcal{S}(\mathcal{H}) := \mathcal{H}'$ des *isomorphismes*

Ceci donne $\mathcal{S}|_{\mathcal{V}}^*(\mathbf{y}) = \mathbf{x}_{\mathcal{V}} + 0 \longrightarrow$ On interpole $\mathcal{S}|_{\mathcal{V}}$ donc \mathcal{V}' avec $36 * 60 / 36 = 60$ signatures

Comme \mathcal{P} est linéaire sur \mathcal{H} , après changement de variable $\mathcal{Q} = \mathcal{P} \circ \mathcal{S}$ sera linéaire sur \mathcal{H}'

Forger une signature UOV

$\mathbb{F}_q^n = \mathcal{V}' + \mathcal{H}'$, donc connaître l'espace \mathcal{V}' ou \mathcal{H}' suffit à forger une signature UOV

On a toujours pas la clef publique...

⁵[Pébereau, 2023]

Récupérer la clef publique

On a 1600 signatures :

- Interpolation polynomiale de la clef publique $\mathcal{Q}_i : \mathbf{y} \mapsto \mathbf{t}_i$
 - $\binom{n+1}{2} = 1891$ inconnues pour chaque \mathcal{Q}_i

Récupérer la clef publique

On a 1600 signatures :

- Interpolation polynomiale de la clef publique $\mathcal{Q}_i : \mathbf{y} \mapsto \mathbf{t}_i$
 - $\binom{n+1}{2} = 1891$ inconnues pour chaque $\mathcal{Q}_i \rightarrow$ n'exploite pas la connaissance de \mathcal{V}'

Récupérer la clef publique

On a 1600 signatures :

- Interpolation polynomiale de la clef publique $\mathcal{Q}_i : \mathbf{y} \mapsto \mathbf{t}_i$
 - $\binom{n+1}{2} = 1891$ inconnues pour chaque $\mathcal{Q}_i \rightarrow$ n'exploite pas la connaissance de \mathcal{V}'
- Interpolation polynomiale de la clef privée, $\mathcal{P}_i : \mathbf{x}_{\mathcal{V}} \oplus \mathbf{x}_{\mathcal{H}} \mapsto \mathbf{t}_i$
 - $((\binom{n+1}{2}) - \binom{h}{2}) = 1591$ inconnues par \mathcal{P}_i

Récupérer la clef publique

On a 1600 signatures :

- Interpolation polynomiale de la clef publique $\mathcal{Q}_i : \mathbf{y} \mapsto \mathbf{t}_i$
 - $\binom{n+1}{2} = 1891$ inconnues pour chaque $\mathcal{Q}_i \rightarrow$ n'exploite pas la connaissance de \mathcal{V}'
- Interpolation polynomiale de la clef privée, $\mathcal{P}_i : \mathbf{x}_{\mathcal{V}} \oplus \mathbf{x}_{\mathcal{H}} \mapsto \mathbf{t}_i$
 - $(\binom{n+1}{2} - \binom{h}{2}) = 1591$ inconnues par $\mathcal{P}_i \rightarrow$ chaque $\mathbf{x}_{\mathcal{H}}$ rajoute des inconnues

Récupérer la clef publique

On a 1600 signatures :

- Interpolation polynomiale de la clef publique $\mathcal{Q}_i : \mathbf{y} \mapsto \mathbf{t}_i$
 - $\binom{n+1}{2} = 1891$ inconnues pour chaque $\mathcal{Q}_i \rightarrow$ n'exploite pas la connaissance de \mathcal{V}'
- Interpolation polynomiale de la clef privée, $\mathcal{P}_i : \mathbf{x}_{\mathcal{V}} \oplus \mathbf{x}_{\mathcal{H}} \mapsto \mathbf{t}_i$
 - $(\binom{n+1}{2} - \binom{h}{2}) = 1591$ inconnues par $\mathcal{P}_i \rightarrow$ chaque $\mathbf{x}_{\mathcal{H}}$ rajoute des inconnues

$$\mathcal{Q}_i(\mathbf{y}) = \mathcal{Q}_i(\mathbf{y}_{\mathcal{V}'} + \mathbf{y}_{\mathcal{H}'}) = \mathcal{Q}_i(\mathbf{y}_{\mathcal{V}'}) + \mathcal{Q}_i(\mathbf{y}_{\mathcal{H}'}) + \mathcal{Q}'_i(\mathbf{y}_{\mathcal{V}'}, \mathbf{y}_{\mathcal{H}'}) - \mathcal{Q}_i(\mathbf{0})$$

Récupérer la clef publique

On a 1600 signatures :

- Interpolation polynomiale de la clef publique $\mathcal{Q}_i : \mathbf{y} \mapsto \mathbf{t}_i$
 - $\binom{n+1}{2} = 1891$ inconnues pour chaque $\mathcal{Q}_i \rightarrow$ n'exploite pas la connaissance de \mathcal{V}'
- Interpolation polynomiale de la clef privée, $\mathcal{P}_i : \mathbf{x}_{\mathcal{V}} \oplus \mathbf{x}_{\mathcal{H}} \mapsto \mathbf{t}_i$
 - $(\binom{n+1}{2} - \binom{h}{2}) = 1591$ inconnues par $\mathcal{P}_i \rightarrow$ chaque $\mathbf{x}_{\mathcal{H}}$ rajoute des inconnues

$$\mathcal{Q}_i(\mathbf{y}) = \mathcal{Q}_i(\mathbf{y}_{\mathcal{V}'} + \mathbf{y}_{\mathcal{H}'}) = \mathcal{Q}_i(\mathbf{y}_{\mathcal{V}'}) + \mathcal{Q}_i(\mathbf{y}_{\mathcal{H}'}) + \mathcal{Q}'_i(\mathbf{y}_{\mathcal{V}'}, \mathbf{y}_{\mathcal{H}'}) - \mathcal{Q}_i(\mathbf{0})$$

↑ ↑ ↑ ↑
1893 inconnues $\binom{37}{2}$ [quad] $\binom{25}{2}$ [quad] $24 \cdot 36$ [biquad] 1 [const]

Récupérer la clef publique

On a 1600 signatures :

- Interpolation polynomiale de la clef publique $\mathcal{Q}_i : \mathbf{y} \mapsto \mathbf{t}_i$
 - $\binom{n+1}{2} = 1891$ inconnues pour chaque $\mathcal{Q}_i \rightarrow$ n'exploite pas la connaissance de \mathcal{V}'
- Interpolation polynomiale de la clef privée, $\mathcal{P}_i : \mathbf{x}_{\mathcal{V}} \oplus \mathbf{x}_{\mathcal{H}} \mapsto \mathbf{t}_i$
 - $(\binom{n+1}{2} - \binom{h}{2}) = 1591$ inconnues par $\mathcal{P}_i \rightarrow$ chaque $\mathbf{x}_{\mathcal{H}}$ rajoute des inconnues

$$\mathcal{Q}_i(\mathbf{y}) = \mathcal{Q}_i(\mathbf{y}_{\mathcal{V}'} + \mathbf{y}_{\mathcal{H}'}) = \mathcal{Q}_i(\mathbf{y}_{\mathcal{V}'}) + \mathcal{Q}_i(\mathbf{y}_{\mathcal{H}'}) + \mathcal{Q}'_i(\mathbf{y}_{\mathcal{V}'}, \mathbf{y}_{\mathcal{H}'}) - \mathcal{Q}_i(\mathbf{0})$$

\uparrow \uparrow \uparrow \uparrow

1593 inconnues $\binom{37}{2}$ [quad] 25 [linear] $24 \cdot 36$ [biquad] 1 [const]

Récupérer la clef publique

On a 1600 signatures :

- Interpolation polynomiale de la clef publique $\mathcal{Q}_i : \mathbf{y} \mapsto \mathbf{t}_i$
 - $\binom{n+1}{2} = 1891$ inconnues pour chaque $\mathcal{Q}_i \rightarrow$ n'exploite pas la connaissance de \mathcal{V}'
- Interpolation polynomiale de la clef privée, $\mathcal{P}_i : \mathbf{x}_{\mathcal{V}} \oplus \mathbf{x}_{\mathcal{H}} \mapsto \mathbf{t}_i$
 - $(\binom{n+1}{2} - \binom{h}{2}) = 1591$ inconnues par $\mathcal{P}_i \rightarrow$ chaque $\mathbf{x}_{\mathcal{H}}$ rajoute des inconnues

$$\mathcal{Q}_i(\mathbf{y}) = \mathcal{Q}_i(\mathbf{y}_{\mathcal{V}'} + \mathbf{y}_{\mathcal{H}'}) = \mathcal{Q}_i(\mathbf{y}_{\mathcal{V}'}) + \mathcal{Q}_i(\mathbf{y}_{\mathcal{H}'}) + \mathcal{Q}'_i(\mathbf{y}_{\mathcal{V}'}, \mathbf{y}_{\mathcal{H}'}) - \mathcal{Q}_i(\mathbf{0})$$

\uparrow \uparrow \uparrow \uparrow

1593 inconnues $\binom{37}{2}$ [quad] 25 [linear] $24 \cdot 36$ [biquad] 1 [const]

On a donc assez de signatures pour interpoler les \mathcal{Q}_i



GPRS et construction de backdoor

Ne JAMAIS faire confiance à de la crypto propriétaire!



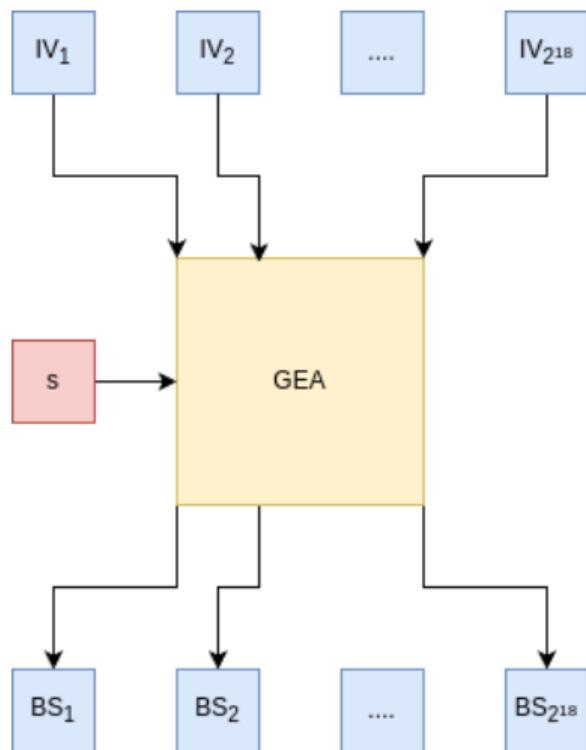
Attacking a cryptographic primitive



Introducing a trapdoor in a cryptographic primitive

- Objectifs :
 - Insérer une backdoor dans un stream-cipher LSFR
- Données :
 - Le code source
 - Sagemath...

Communication avec le serveur



- 2^{18} IVs connus donnent 2^{18} BitStreams connus sur 64 bits
- Environ 10 minutes de génération
- Objectif : retrouver le secret s

GPRS et GEA

GPRS (General Packet Radio Service)

Techno	Débit
2G	30 kbps
GPRS	100 kbps
EDGE	200 kbps
3G	1 Mbps

- Envoi de données par paquets
→ Web, MMS, email amélioré

GPRS et GEA

GPRS (General Packet Radio Service)

Techno	Débit
2G	30 kbps
GPRS	100 kbps
EDGE	200 kbps
3G	1 Mbps

- Envoi de données par paquets
→ Web, MMS, email amélioré

GEA (GPRS Encryption Algorithm)

- Versions GEA1 à GEA5
- ETSI (European Telecommunications Standards Institute), 1998

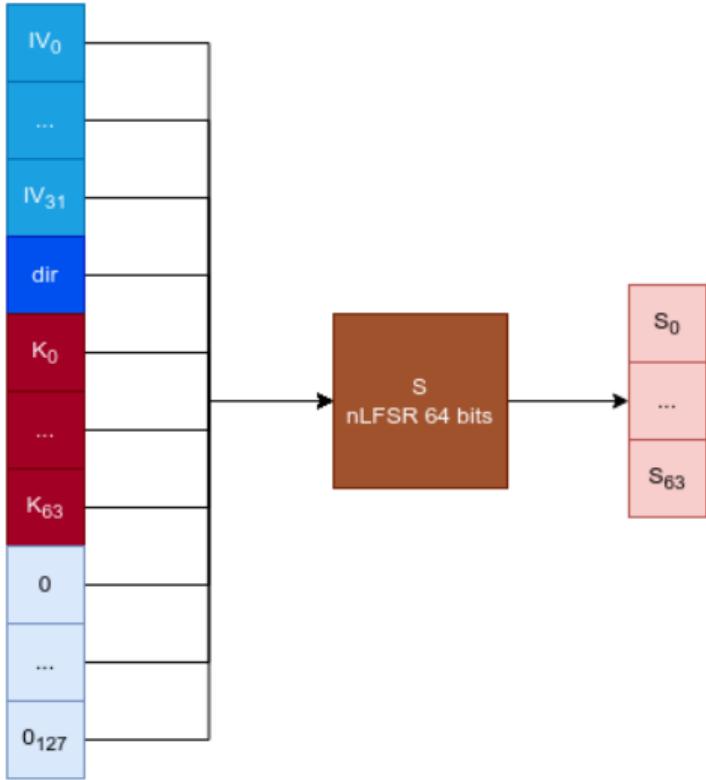
"Doit se conformer aux restrictions européennes en vigueur sur les exports nationaux de la cryptographie"

- GEA1 et GEA2 vulnérables

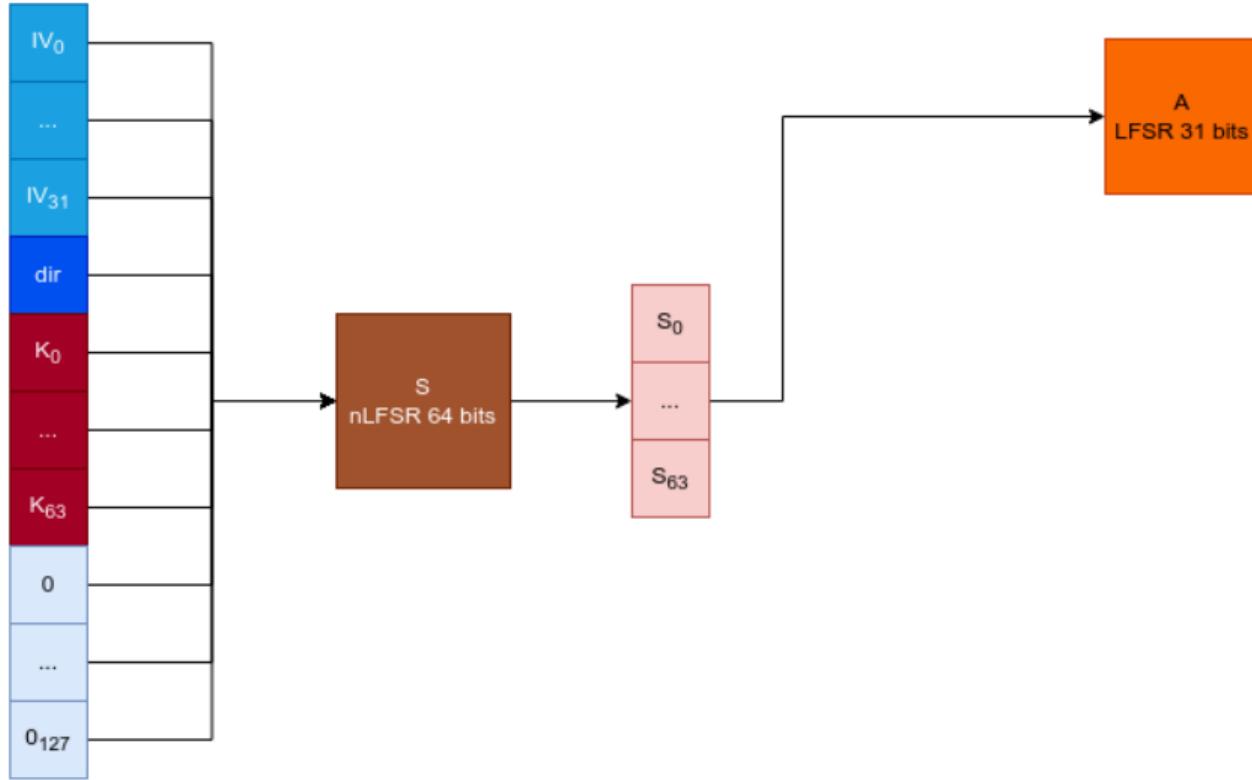
GEA1

IV ₀
...
IV ₃₁
dir
K ₀
...
K ₆₃
0
...
0 ₁₂₇

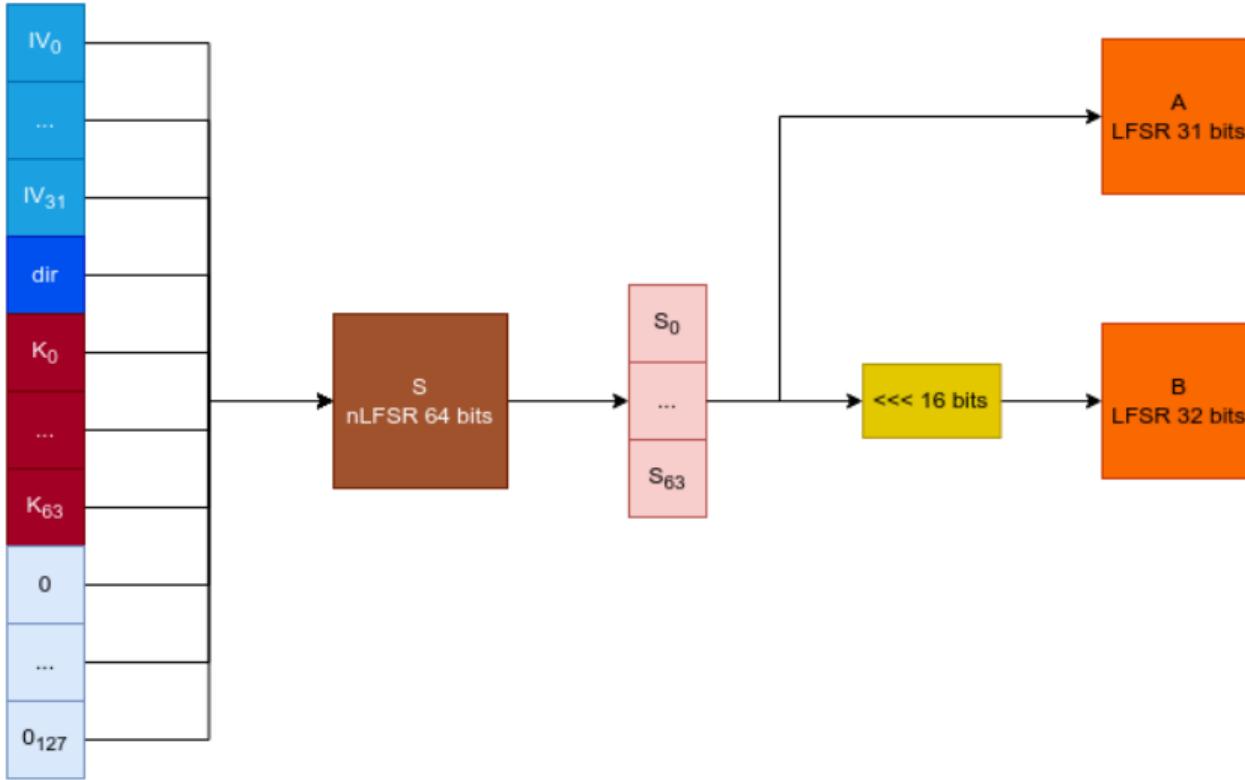
GEA1



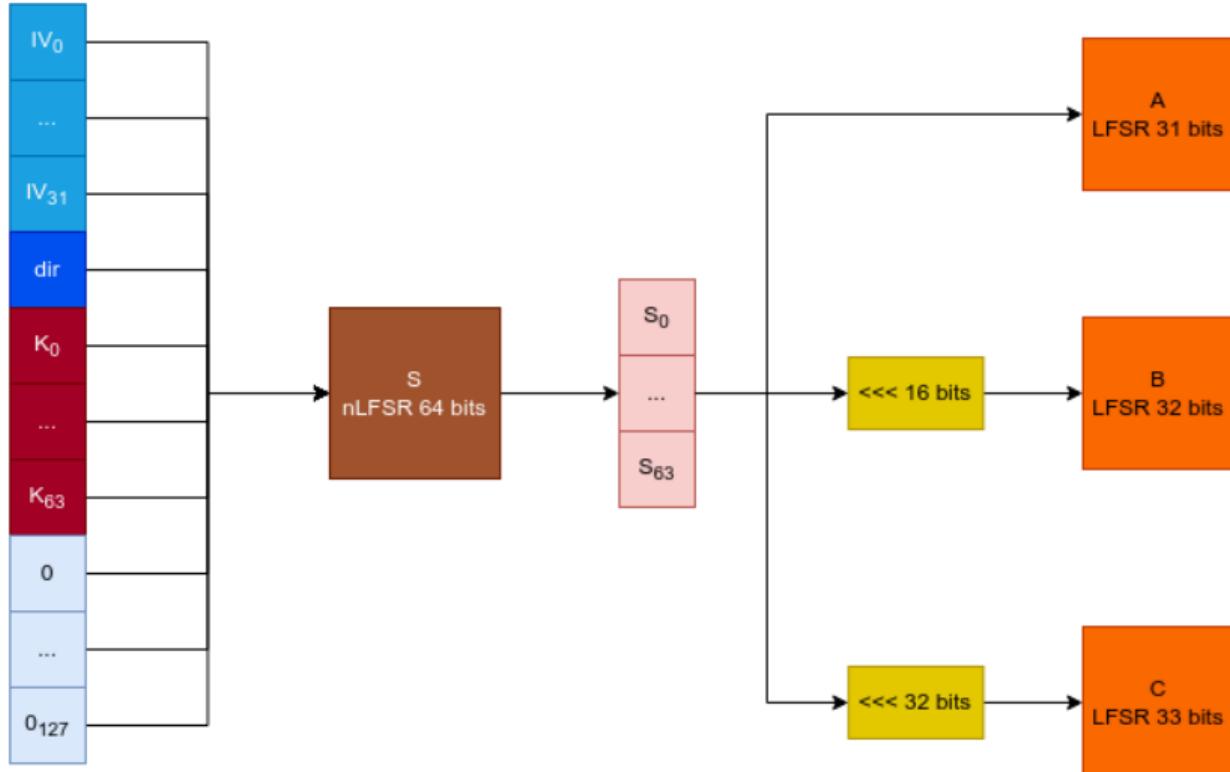
GEA1



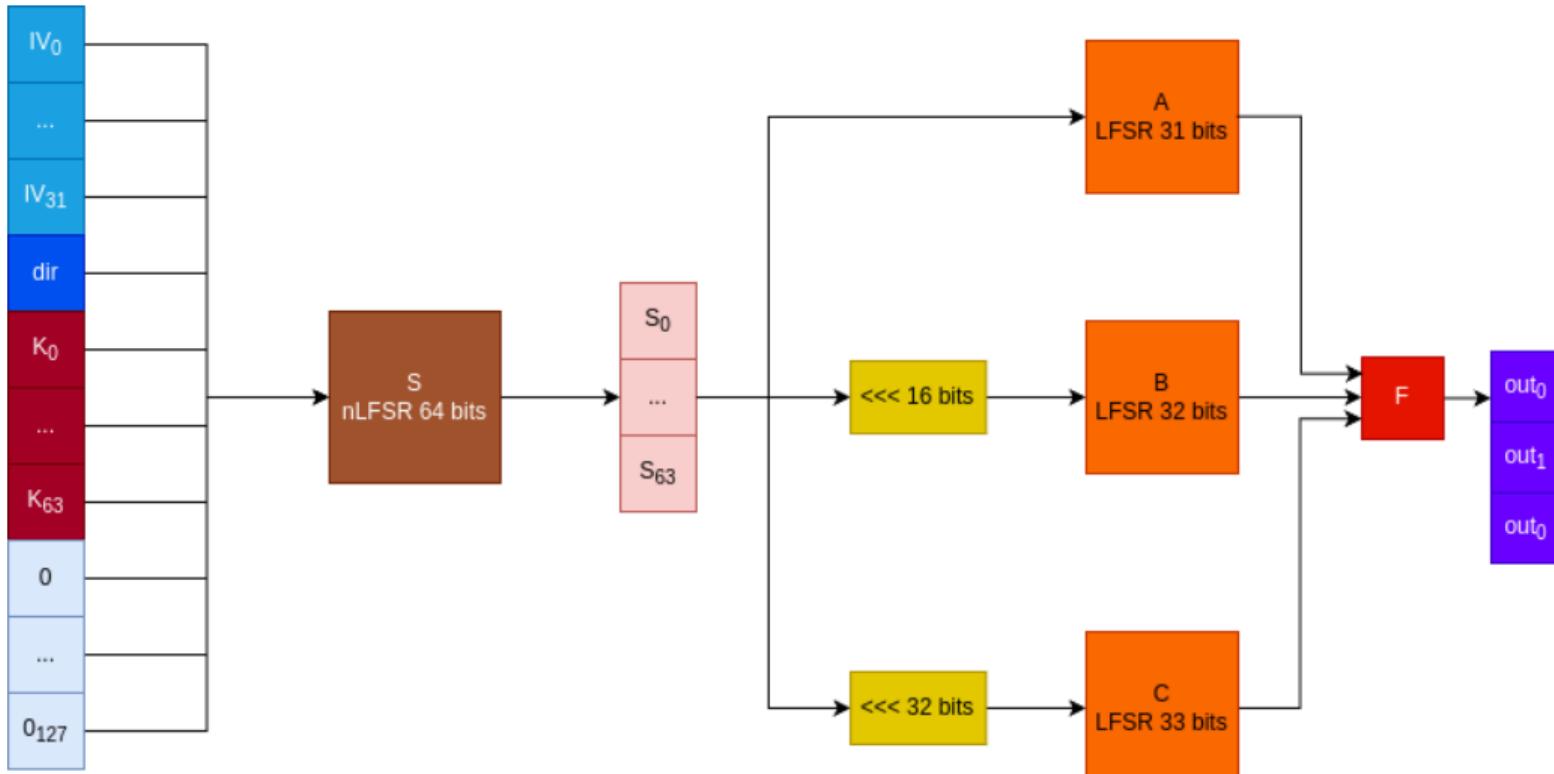
GEA1



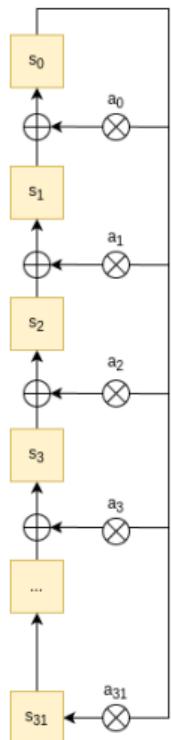
GEA1



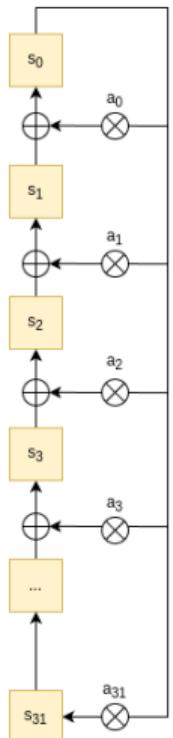
GEA1



Sur les LFSRs de Galois (Linear Feedback Shift Register)



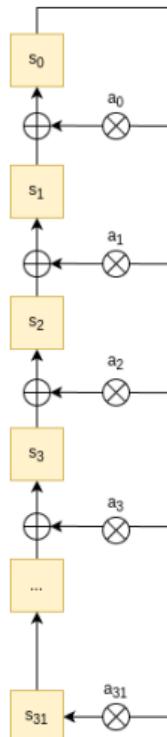
Sur les LFSRs de Galois (Linear Feedback Shift Register)



$$\begin{cases} s_0 \leftarrow s_1 + a_0 s_0 \\ s_1 \leftarrow s_2 + a_1 s_0 \\ \vdots \\ s_{30} \leftarrow s_{31} + a_{30} s_0 \\ s_{31} \leftarrow 0 + a_{31} s_0 \end{cases}$$

$$\begin{pmatrix} s_0 \\ s_1 \\ \vdots \\ s_{30} \\ s_{31} \end{pmatrix} \leftarrow \begin{pmatrix} a_0 & 1 & 0 & \dots & 0 \\ a_1 & 0 & 1 & & 0 \\ \vdots & \vdots & & \ddots & \\ a_{30} & 0 & \dots & 0 & 1 \\ a_{31} & 0 & \dots & 0 & 0 \end{pmatrix} \begin{pmatrix} s_0 \\ s_1 \\ \vdots \\ s_{30} \\ s_{31} \end{pmatrix}$$

Sur les LFSRs de Galois (Linear Feedback Shift Register)

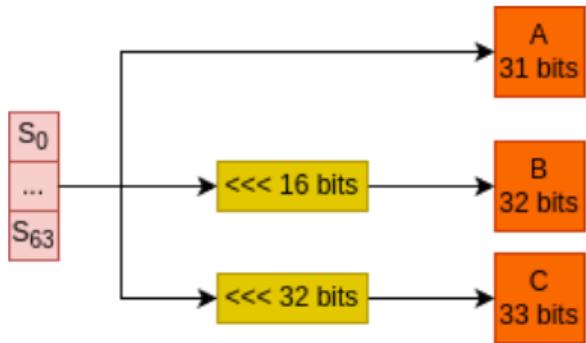


$$\begin{cases} s_0 \leftarrow s_1 + a_0 s_0 \\ s_1 \leftarrow s_2 + a_1 s_0 \\ \vdots \\ s_{30} \leftarrow s_{31} + a_{30} s_0 \\ s_{31} \leftarrow 0 + a_{31} s_0 \end{cases}$$

$$\begin{pmatrix} s_0 \\ s_1 \\ \vdots \\ s_{30} \\ s_{31} \end{pmatrix} \leftarrow \begin{pmatrix} a_0 & 1 & 0 & \dots & 0 \\ a_1 & 0 & 1 & & 0 \\ \vdots & \vdots & & \ddots & \\ a_{30} & 0 & \dots & 0 & 1 \\ a_{31} & 0 & \dots & 0 & 0 \end{pmatrix} \begin{pmatrix} s_0 \\ s_1 \\ \vdots \\ s_{30} \\ s_{31} \end{pmatrix}$$

- Fait intervenir la matrice compagnon G_P de $P = X^{32} + a_{31}X^{31} \dots + a_1X + a_0$

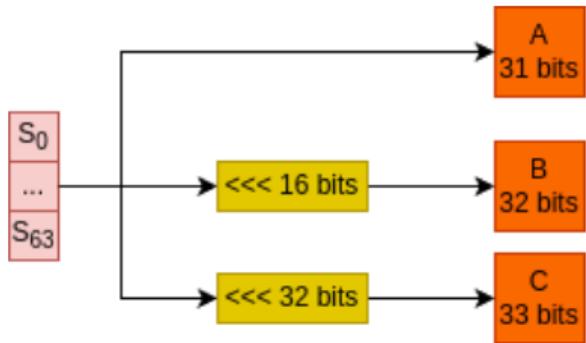
Dans le cadre de GEA1



- L'état initial de A, B, C est linéaire en l'entrée secrète s
- On a les états $M_A.s$, $M_B.s$, $M_C.s$ des matrices de taille 31×64 , 32×64 , 33×64

⁶[Beierle et al., 2021b]

Dans le cadre de GEA1



- L'état initial de A, B, C est linéaire en l'entrée secrète s
- On a les états $\mathbf{M}_A.s$, $\mathbf{M}_B.s$, $\mathbf{M}_C.s$ des matrices de taille 31×64 , 32×64 , 33×64
- $T_{AC} := \ker(\mathbf{M}_A) \cap \ker(\mathbf{M}_C)$ est de dimension 24 et $T_{AC} \cap \ker(\mathbf{M}_B) = \{0\}$
- Ceci donne une récupération en 2^{37} appels d'oracle, 2^{40} bruteforces, en utilisant 2^8 tables de $2^{24} \times 89$ bits (48 Go)⁶

⁶[Beierle et al., 2021b]

Et dans le cas du challenge?

B est rotate de 21 (et non 16) et C est rotate de 42 (et non 32)

- Quels polynômes donnent une attaque?

⁷[Beierle et al., 2021a]

Et dans le cas du challenge?

B est rotate de 21 (et non 16) et C est rotate de 42 (et non 32)

- Quels polynômes donnent une attaque?
- Comment les polynômes de GEA1 ont été trouvés? ⁷

⁷[Beierle et al., 2021a]

Et dans le cas du challenge?

B est rotate de 21 (et non 16) et C est rotate de 42 (et non 32)

- Quels polynômes donnent une attaque?
- Comment les polynômes de GEA1 ont été trouvés? ⁷

Condition d'appartenance au kernel après rotation

Si $\mathbf{t} = (t_0, \dots, t_{63}) \in \ker(\mathbf{M}_A)$ où A est un LFSR de polynôme de Galois primitif P_A dont l'entrée est rotation de k , alors

$$P_A \mid (t_0X^k + \dots + t_{k-1}X^1) + (t_kX^{64} + \dots + t_{63}X^{k+1}) = \sum_{i=0}^{k-1} t_i X^{k-i} + \sum_{i=k}^{63} t_i X^{64-i+k}$$

Par exemple, en l'absence de rotation, $P_A \mid \sum_{i=0}^{63} t_i X^{64-i}$

⁷[Beierle et al., 2021a]

Partage de kernel

On va plutôt partir du kernel et chercher des polynômes qui ont ce kernel en commun ⁸

Condition d'appartenance au kernel après rotation

Soit deux LSFR A et B. A n'est pas shifté et B est shifté de k. Si
 $\mathbf{t} = (t_0, \dots, t_{63}) \in \ker(\mathbf{M}_A) \cap \ker(\mathbf{M}_B) := T_{A,B}$ alors

$$P_A \mid \sum_{i=0}^{63} t_i X^{64-i} \text{ et } P_B \mid \sum_{i=0}^{k-1} t_i X^{k-i} + \sum_{i=k}^{63} t_i X^{64-i+k}$$

De plus, $\dim(T_{A,B}) \geq r$ où X^r divise les deux polynômes de droite

⁸[Beierle et al., 2021a]

Partage de kernel

On va plutôt partir du kernel et chercher des polynômes qui ont ce kernel en commun ⁸

Condition d'appartenance au kernel après rotation

Soit deux LSFR A et B. A n'est pas shifté et B est shifté de k. Si
 $\mathbf{t} = (t_0, \dots, t_{63}) \in \ker(\mathbf{M}_A) \cap \ker(\mathbf{M}_B) := T_{A,B}$ alors

$$P_A \mid \sum_{i=0}^{63} t_i X^{64-i} \text{ et } P_B \mid \sum_{i=0}^{k-1} t_i X^{k-i} + \sum_{i=k}^{63} t_i X^{64-i+k}$$

De plus, $\dim(T_{A,B}) \geq r$ où X^r divise les deux polynômes de droite
car alors $\mathbf{t}_i = (0, \dots, 0, t_0, \dots, t_{63-i}) \in T_{A,B}$ pour $i < r$

⁸[Beierle et al., 2021a]

Un exemple jouet

- Si B est shifté de 32 et que l'on cherche $T_{A,B}$ de dimension 31, on cherche \mathbf{t} tel que

$$X^{31} \mid \sum_{i=0}^{63} t_i X^{64-i} \text{ et } X^{31} \mid \sum_{i=0}^{32-1} t_i X^{32-i} + \sum_{i=32}^{63} t_i X^{64-i+32}$$

Un exemple jouet

- Si B est shifté de 32 et que l'on cherche $T_{A,B}$ de dimension 31, on cherche \mathbf{t} tel que

$$X^{31} \mid \sum_{i=0}^{63} t_i X^{64-i} \text{ et } X^{31} \mid \sum_{i=0}^{32-1} t_i X^{32-i} + \sum_{i=32}^{63} t_i X^{64-i+32}$$

- L'équation gauche force $t_{63} \mapsto t_{34}$ à zéro. L'équation droite force $t_1 \mapsto t_{31}$ à zéro.
 - Brute-force exhaustif de t_0, t_{32} et t_{33} (8 solutions)

Un exemple jouet

- Si B est shifté de 32 et que l'on cherche $T_{A,B}$ de dimension 31, on cherche \mathbf{t} tel que

$$X^{31} \mid \sum_{i=0}^{63} t_i X^{64-i} \text{ et } X^{31} \mid \sum_{i=0}^{32-1} t_i X^{32-i} + \sum_{i=32}^{63} t_i X^{64-i+32}$$

- L'équation gauche force $t_{63} \mapsto t_{34}$ à zéro. L'équation droite force $t_1 \mapsto t_{31}$ à zéro.
 - Brute-force exhaustif de t_0, t_{32} et t_{33} (8 solutions)
- On cherche alors
 - $P_A \mid t_0 X^{64} + t_{32} X^{32} + t_{33} X^{31}$
 - $P_B \mid t_{32} X^{64} + t_{33} X^{63} + t_0 X^{32}$

Un exemple jouet

- Si B est shifté de 32 et que l'on cherche $T_{A,B}$ de dimension 31, on cherche \mathbf{t} tel que

$$X^{31} \mid \sum_{i=0}^{63} t_i X^{64-i} \text{ et } X^{31} \mid \sum_{i=0}^{32-1} t_i X^{32-i} + \sum_{i=32}^{63} t_i X^{64-i+32}$$

- L'équation gauche force $t_{63} \mapsto t_{34}$ à zéro. L'équation droite force $t_1 \mapsto t_{31}$ à zéro.
 - Brute-force exhaustif de t_0, t_{32} et t_{33} (8 solutions)
 - On cherche alors
 - $P_A \mid t_0 X^{64} + t_{32} X^{32} + t_{33} X^{31}$
 - $P_B \mid t_{32} X^{64} + t_{33} X^{63} + t_0 X^{32}$
 - Tous polynômes **primitifs de la bonne taille** produiront le bon kernel
- ⇒ La dimension du kernel est bornée par le shift **relatif**

Exploiter les shifts

- GEA1 : Shifts 16 et 32 \Rightarrow kernel joint maximal de dimension 32 entre A et C

Exploiter les shifts

- GEA1 : Shifts 16 et 32 \Rightarrow kernel joint maximal de dimension 32 entre A et C
- CTF : Shifts 21 et 42 \Rightarrow kernel joint maximal de dimension 22 entre C et A

Exploiter les shifts

- GEA1 : Shifts 16 et 32 \Rightarrow kernel joint maximal de dimension 32 entre A et C
- CTF : Shifts 21 et 42 \Rightarrow kernel joint maximal de dimension 22 entre C et A
- Exploiter la symétrie, existe-t-il un $T_{A,B,C}$ grand?

Exploiter les shifts

- GEA1 : Shifts 16 et 32 \Rightarrow kernel joint maximal de dimension 32 entre A et C
- CTF : Shifts 21 et 42 \Rightarrow kernel joint maximal de dimension 22 entre C et A
- Exploiter la symétrie, existe-t-il un $T_{A,B,C}$ grand?

OUI! On trouve par bruteforce un $T_{A,B,C}$ de dimension 16

Exploiter les shifts

- GEA1 : Shifts 16 et 32 \Rightarrow kernel joint maximal de dimension 32 entre A et C
- CTF : Shifts 21 et 42 \Rightarrow kernel joint maximal de dimension 22 entre C et A
- Exploiter la symétrie, existe-t-il un $T_{A,B,C}$ grand?

OUI! On trouve par bruteforce un $T_{A,B,C}$ de dimension 16

- $f = X^{31} + X^{27} + X^{26} + X^{25} + X^{23} + X^{17} + X^{15} + X^{13} + X^{11} + X^9 + X^7 + X^6 + X^5 + X^3 + X^2 + X + 1$
- $g = X^{32} + X^{30} + X^{27} + X^{23} + X^{18} + X^{17} + X^{16} + X^{12} + X^{11} + X^9 + X^8 + X^6 + X^4 + X^3 + X^2 + X + 1$
- $h = X^{33} + X^{29} + X^{28} + X^{27} + X^{25} + X^{24} + X^{23} + X^{22} + X^{20} + X^{19} + X^{18} + X^{17} + X^9 + X^6 + X^5 + X^4 + X^3 + X + 1$

Exploiter les shifts

- GEA1 : Shifts 16 et 32 \Rightarrow kernel joint maximal de dimension 32 entre A et C
- CTF : Shifts 21 et 42 \Rightarrow kernel joint maximal de dimension 22 entre C et A
- Exploiter la symétrie, existe-t-il un $T_{A,B,C}$ grand?

OUI! On trouve par bruteforce un $T_{A,B,C}$ de dimension 16

- $f = X^{31} + X^{27} + X^{26} + X^{25} + X^{23} + X^{17} + X^{15} + X^{13} + X^{11} + X^9 + X^7 + X^6 + X^5 + X^3 + X^2 + X + 1$
- $g = X^{32} + X^{30} + X^{27} + X^{23} + X^{18} + X^{17} + X^{16} + X^{12} + X^{11} + X^9 + X^8 + X^6 + X^4 + X^3 + X^2 + X + 1$
- $h = X^{33} + X^{29} + X^{28} + X^{27} + X^{25} + X^{24} + X^{23} + X^{22} + X^{20} + X^{19} + X^{18} + X^{17} + X^9 + X^6 + X^5 + X^4 + X^3 + X + 1$

$$\mathbb{F}_2^{64} = T_{A,B,C} \oplus V \text{ et donc } \mathbf{s} = \mathbf{s}_T + \mathbf{s}_V$$

Aussi, $\mathbf{s} \mapsto (\mathbf{M}_A \cdot \mathbf{s}, \mathbf{M}_B \cdot \mathbf{s}, \mathbf{M}_C \cdot \mathbf{s}) = (\mathbf{M}_A \cdot \mathbf{s}_V, \mathbf{M}_B \cdot \mathbf{s}_V, \mathbf{M}_C \cdot \mathbf{s}_V)$ n'a que 48 bits d'entropie

Attaque finale

- Objectif : Grace aux 2^{18} bitstreams , trouver l'un des **s**, pour revenir à la clef

Attaque finale

- Objectif : Grace aux 2^{18} bitstreams , trouver l'un des \mathbf{s} , pour revenir à la clef
- $\mathbb{F}_2^{64} = T_{A,B,C} \oplus V = T_{A,B,C} \oplus X \oplus Y$ et donc $\mathbf{s} = \mathbf{s}_T + \mathbf{s}_X + \mathbf{s}_Y$
 - X de dimension 30 et Y de dimension 18

Attaque finale

- Objectif : Grace aux 2^{18} bitstreams , trouver l'un des \mathbf{s} , pour revenir à la clef
- $\mathbb{F}_2^{64} = T_{A,B,C} \oplus V = T_{A,B,C} \oplus X \oplus Y$ et donc $\mathbf{s} = \mathbf{s}_T + \mathbf{s}_X + \mathbf{s}_Y$
 - X de dimension 30 et Y de dimension 18
 - Bitstream $BS(\mathbf{s}) = BS(\mathbf{s}_X + \mathbf{s}_Y)$

Attaque finale

- Objectif : Grace aux 2^{18} bitstreams , trouver l'un des \mathbf{s} , pour revenir à la clef
- $\mathbb{F}_2^{64} = T_{A,B,C} \oplus V = T_{A,B,C} \oplus X \oplus Y$ et donc $\mathbf{s} = \mathbf{s}_T + \mathbf{s}_X + \mathbf{s}_Y$
 - X de dimension 30 et Y de dimension 18
 - Bitstream $BS(\mathbf{s}) = BS(\mathbf{s}_X + \mathbf{s}_Y)$
- Précalcul des 2^{30} valeurs de $BS(\mathbf{s}_X + 0)$ dans une hashmap H

Attaque finale

- Objectif : Grace aux 2^{18} bitstreams , trouver l'un des \mathbf{s} , pour revenir à la clef
- $\mathbb{F}_2^{64} = T_{A,B,C} \oplus V = T_{A,B,C} \oplus X \oplus Y$ et donc $\mathbf{s} = \mathbf{s}_T + \mathbf{s}_X + \mathbf{s}_Y$
 - X de dimension 30 et Y de dimension 18
 - Bitstream $BS(\mathbf{s}) = BS(\mathbf{s}_X + \mathbf{s}_Y)$
- Précalcul des 2^{30} valeurs de $BS(\mathbf{s}_X + 0)$ dans une hashmap H
 - $H : BS(64 \text{ bits}) \mapsto \mathbf{s}_X(30 \text{ bits})$

Attaque finale

- Objectif : Grace aux 2^{18} bitstreams , trouver l'un des \mathbf{s} , pour revenir à la clef
- $\mathbb{F}_2^{64} = T_{A,B,C} \oplus V = T_{A,B,C} \oplus X \oplus Y$ et donc $\mathbf{s} = \mathbf{s}_T + \mathbf{s}_X + \mathbf{s}_Y$
 - X de dimension 30 et Y de dimension 18
 - Bitstream $BS(\mathbf{s}) = BS(\mathbf{s}_X + \mathbf{s}_Y)$
- Précalcul des 2^{30} valeurs de $BS(\mathbf{s}_X + 0)$ dans une hashmap H
 - $H : BS(64 \text{ bits}) \mapsto \mathbf{s}_X(30 \text{ bits})$
 - Un bitstream fait 64 bits $\Rightarrow 64 * 2^{30} = 8\text{Go}$ de stockage

Attaque finale

- Objectif : Grace aux 2^{18} bitstreams , trouver l'un des \mathbf{s} , pour revenir à la clef
- $\mathbb{F}_2^{64} = T_{A,B,C} \oplus V = T_{A,B,C} \oplus X \oplus Y$ et donc $\mathbf{s} = \mathbf{s}_T + \mathbf{s}_X + \mathbf{s}_Y$
 - X de dimension 30 et Y de dimension 18
 - Bitstream $BS(\mathbf{s}) = BS(\mathbf{s}_X + \mathbf{s}_Y)$
- Précalcul des 2^{30} valeurs de $BS(\mathbf{s}_X + 0)$ dans une hashmap H
 - $H : BS(64 \text{ bits}) \mapsto \mathbf{s}_X(30 \text{ bits})$
 - Un bitstream fait 64 bits $\Rightarrow 64 * 2^{30} = 8\text{Go}$ de stockage
- Statistiquement, l'un des $\mathbf{s}_Y = 0$

Attaque finale

- Objectif : Grace aux 2^{18} bitstreams , trouver l'un des \mathbf{s} , pour revenir à la clef
- $\mathbb{F}_2^{64} = T_{A,B,C} \oplus V = T_{A,B,C} \oplus X \oplus Y$ et donc $\mathbf{s} = \mathbf{s}_T + \mathbf{s}_X + \mathbf{s}_Y$
 - X de dimension 30 et Y de dimension 18
 - Bitstream $BS(\mathbf{s}) = BS(\mathbf{s}_X + \mathbf{s}_Y)$
- Précalcul des 2^{30} valeurs de $BS(\mathbf{s}_X + 0)$ dans une hashmap H
 - $H : BS(64 \text{ bits}) \mapsto \mathbf{s}_X(30 \text{ bits})$
 - Un bitstream fait 64 bits $\Rightarrow 64 * 2^{30} = 8\text{Go}$ de stockage
- Statistiquement, l'un des $\mathbf{s}_Y = 0$
 - Le $BS(\mathbf{s})$ associé est dans H

Attaque finale

- Objectif : Grace aux 2^{18} bitstreams , trouver l'un des \mathbf{s} , pour revenir à la clef
- $\mathbb{F}_2^{64} = T_{A,B,C} \oplus V = T_{A,B,C} \oplus X \oplus Y$ et donc $\mathbf{s} = \mathbf{s}_T + \mathbf{s}_X + \mathbf{s}_Y$
 - X de dimension 30 et Y de dimension 18
 - Bitstream $BS(\mathbf{s}) = BS(\mathbf{s}_X + \mathbf{s}_Y)$
- Précalcul des 2^{30} valeurs de $BS(\mathbf{s}_X + 0)$ dans une hashmap H
 - $H : BS(64 \text{ bits}) \mapsto \mathbf{s}_X(30 \text{ bits})$
 - Un bitstream fait 64 bits $\Rightarrow 64 * 2^{30} = 8\text{Go}$ de stockage
- Statistiquement, l'un des $\mathbf{s}_Y = 0$
 - Le $BS(\mathbf{s})$ associé est dans H
 - H nous donne \mathbf{s}_X

Attaque finale

- Objectif : Grace aux 2^{18} bitstreams , trouver l'un des \mathbf{s} , pour revenir à la clef
- $\mathbb{F}_2^{64} = T_{A,B,C} \oplus V = T_{A,B,C} \oplus X \oplus Y$ et donc $\mathbf{s} = \mathbf{s}_T + \mathbf{s}_X + \mathbf{s}_Y$
 - X de dimension 30 et Y de dimension 18
 - Bitstream $BS(\mathbf{s}) = BS(\mathbf{s}_X + \mathbf{s}_Y)$
- Précalcul des 2^{30} valeurs de $BS(\mathbf{s}_X + 0)$ dans une hashmap H
 - $H : BS(64 \text{ bits}) \mapsto \mathbf{s}_X(30 \text{ bits})$
 - Un bitstream fait 64 bits $\Rightarrow 64 * 2^{30} = 8\text{Go}$ de stockage
- Statistiquement, l'un des $\mathbf{s}_Y = 0$
 - Le $BS(\mathbf{s})$ associé est dans H
 - H nous donne \mathbf{s}_X
- On retrouve $\mathbf{s} = \mathbf{s}_T + \mathbf{s}_X + 0$ (2^{16} valeurs de \mathbf{s}_T)
- Avec \mathbf{s} et **IV** associé, on retrouve la clef
- Total : $2^{30} + 2^{16}$ opérations, 8Go de mémoire



Merci
RDV en 2026 :) ?

Références

-  Beierle, C., Beyne, T., Felke, P., and Leander, G. (2021a).
Constructing and deconstructing intentional weaknesses in symmetric ciphers.
Cryptology ePrint Archive, Paper 2021/829.
-  Beierle, C., Derbez, P., Leander, G., Leurent, G., Raddum, H., Rotella, Y., Rupprecht, D., and Stennes, L. (2021b).
Cryptanalysis of the GPRS encryption algorithms GEA-1 and GEA-2.
Cryptology ePrint Archive, Paper 2021/819.
-  de Micheli, G. and Heninger, N. (2020).
Recovering cryptographic keys from partial information, by example.
working paper or preprint.
-  Jean, J. (2016).
TikZ for Cryptographers.
<https://www.iacr.org/authors/tikz/>.
-  Leurent, G. and Peyrin, T. (2020).
Sha-1 is a shambles: first chosen-prefix collision on sha-1 and application to the pgp web of trust.
In *Proceedings of the 29th USENIX Conference on Security Symposium*, SEC'20, USA. USENIX