Rapport Projet Cryptanalyse

Louis Coumau

8 Décembre 2020

Table des matières

1 Cryptanalyse de A5/2

1.1 Question 1

Création des fonction :

1. `simul(R,P)`
2. `maj(a,b,c)`
3. `a5_2_init(K,IV)`
4. `a5_2_step(maj)`
5. `a5_2_exec(N,maj)`

Voir code sage.

1.2 Question 2

On veut montrer que, après chaque étape de la production de la suite chiffrante, les valeurs des registres peuvent s'exprimer au moyen d'équations linéaires en les x_i, c'est à dire qu'aucun des monômes des équations peuvent être de la forme $x_i \times x_j$ donc de degré supérieur à 1.
Or, lors de l'exécution de la fonction `a5_2_step()`, les registres sont simplement mis à jour par un LFSR suivant les valeurs de `R4`, ainsi, on ne pourra jamais avoir un monome de la forme $x_i \times x_j$ dans un des registres.

1.3 Question 3

Création des fonction :

1. `create_LFSR_matrix(P)`
2. `a5_2_step_Q3()`
3. `a5_2_exec_Q3(N)`

Voir code sage.

1.4 Question 4

Création des fonction :

1. `algebric_expr(sup)`

Voir code sage.
Forme algébrique de `maj()` : $f(x_1, x_2, x_3) = x_1x_2 \oplus x_1x_3 \oplus x_2x_3$

1.5 Question 5

Voir code sage.

1.6 Question 6

Création des fonction :

1. `sum_arith_prog(n)`
2. `create_monom_list(value_list)`

Voir code sage pour la liste M.

Pour dénombrer le nombre de monômes possibles, intéressons nous à la façon dont sont fabriqué les éléments de la suite chiffrante :

On a z_i un élément de la suite chiffrante. La fonction `a5_2_step()` nous indique que $z_i = y_1 + y_1 + y_3$ avec

$$y_1 = R_{1,0} + maj(R_{1,3}, R_{1,4} \oplus 1, R_{1,6})$$

$$y_2 = R_{2,0} + maj(R_{2,8}, R_{2,5} \oplus 1, R_{2,12})$$

$$y_3 = R_{3,0} + maj(R_{3,4}, R_{3,9} \oplus 1, R_{3,6})$$

Pour y_1, `maj()` ne prend que des valeurs du registre R_1, ainsi, on va compter le nombre de monômes possible avec les valeurs du registre R_1. Ce qui nous donne :

$$\binom{19}{2} = 1 + 2 + 3 + ... + (19 - 2) + (19 - 1) + 19$$

Or, nous somme dans $\mathbb{F}_2$, il ne peux pas y avoir de carré x_i^2 on a donc :

$$1 + 2 + 3 + ... + (19 - 2) + (19 - 1) = 171$$

Même raisonnement avec R_2 et R_3 :

$$1 + 2 + 3 + ... + (22 - 2) + (22 - 1) = 231$$

$$1 + 2 + 3 + ... + (23 - 2) + (23 - 1) = 253$$

Ainsi, on obtient :

$$171 + 231 + 253 = 655$$

1.7 Question 7

Création des fonction :

1. `create_dico(M)`
2. `create_matrix(list_of_equations,M)`

Voir code sage pour les commentaires des fonctions.

1.8 Question 8

Voir code sage pour les commentaires des fonctions.

On souhaite déterminer le contenue des registres R_1, R_2 et R_3. On va procéder de la façon suivante :

- Initialisation :
  - On a z la suite chiffrante
  - On définit les registres :
    - $R_1 1 = (x_0, \ldots, x_{18})$
    - $R_2 = (x_{19}, \ldots, x_{40})$
    - $R_3 = (x_{41}, \ldots, x_{63})$
    - $R_{1,3} = 1$, $R_{2,5} = 1$, $R_{3,4} = 1$
- Exécution : On génère ez la liste des 700 équations de la suite chiffrante avec la fonction `a5_2_exec(700,fmaj)` avec `fmaj()` la forme algébrique de la fonction `maj()`
- Linearisation : On défini L la matrice de linéarisation avec la fonction `create_matrix(ez,M)` dont M est la liste des 655 monômes possibles. La fonction nous renvoie aussi la vecteur V des 1 existant dans les équations.
- Résolution : On résout l'équation suivante :

$$L \times vect(M) = (vect(z) + vect(V))$$

avec le code suivant :
`Solution=LinearMatrix.solve_right(vector(z)+Vector)`
On obtient ainsi les valeurs des monômes dans l'ordre de la liste M.

La solution obtenue est donc la valeur des monômes dans l'ordre de la liste M. On rappelle que le but est de déterminer les valeurs des x_i des registres. On va donc retrouver les valeurs des monômes de degré 1.

- Récupération : On va récupérer l'indice des monômes de degré 1 dans M. (On a dans M, un ordre des monômes de degré 1 croissant, c'est à dire que l'indice de x_0 est inférieur à l'indice de x_1 etc..)
  On crée une fonction `get_value_sol()` permettant de récupérer les valeurs des monômes en fonction de leur position. Cette fonction va nous renvoyer `Bin_sol`, la liste de ces solutions.
- Reconstruction : Ainsi, les 18 premières valeurs de cette liste sont les valeurs des variables de x_0 à x_{18}. Or, on rappelle que 3 valeurs qui ont été fixées à 1 ne sont pas présentes, soit x_3, dans le cas de R_1.
  - `R1=Bin_sol[0:18]`

— `R2=Bin_sol[18:39]`
— `R3=Bin_sol[39:61]`

Puis on insère les 1 à la bonne position :

— `R1.insert(3, 1)`
— `R1.insert(5, 1)`
— `R1.insert(4, 1)`

On obtient :

```
R1= [1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0]
R2= [1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1]
R3= [1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0]
```

On peut maintenant vérifier si les registres définit avec ce contenue génère bien z :

— Vérification : `z==a5_2_exec(700,fmaj)`
— Alternative : Pour retrouver les valeurs des registres, on aurait aussi pu utiliser une base de Grobner :

```
I=Ideal(equations_of_z[i]+z[i] for i in range(700))
gb = I.groebner_basis()
Ks=[el.constant_coefficient() for el in gb ]
print(Ks)
```

Ce qui nous renvoie :

```
[1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0,
1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1,
1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0]
```

Effectivement, on trouve les registres que l'on à trouvé, sans les valeurs à 1, c'est a dire x_3, x_{5+19} et x_{4+41}, il suffirait donc de les insérer.

1.9 Question 9

Après la première boucle On va procéder par récurrence forte :

On va considérer un tour de boucle à l'étape j (et non pas i pour rester cohérent avec la formule.)

On commence avec $X = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} = R^{(0)}$

— L'étape $j = 0$:

— Mise à jour du registre :

$$X \leftarrow A_1 R^{(0)} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$$

— On ajoute K_0 à la dernière position :

$$X \leftarrow X + \begin{pmatrix} 0 \\ \vdots \\ 0 \\ K_0 \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ K_0 \end{pmatrix}$$

— L'étape $j = 1$
— Mise à jour du registre :

$$X \leftarrow A_1 X = A_1 \begin{pmatrix} 0 \\ \vdots \\ 0 \\ K_0 \end{pmatrix}$$

— On ajoute K_1 à la dernière position :

$$X \leftarrow X + \begin{pmatrix} 0 \\ \vdots \\ 0 \\ K_1 \end{pmatrix} = A_1 \begin{pmatrix} 0 \\ \vdots \\ 0 \\ K_0 \end{pmatrix} + \begin{pmatrix} 0 \\ \vdots \\ 0 \\ K_1 \end{pmatrix}$$

— L'étape $j = 2$
— Mise à jour du registre :

$$\begin{aligned} X \leftarrow A_1 X &= A_1 \left[A_1 \begin{pmatrix} 0 \\ \vdots \\ 0 \\ K_0 \end{pmatrix} + \begin{pmatrix} 0 \\ \vdots \\ 0 \\ K_1 \end{pmatrix} \right] \\ &= A_1^2 \begin{pmatrix} 0 \\ \vdots \\ 0 \\ K_0 \end{pmatrix} + A_1 \begin{pmatrix} 0 \\ \vdots \\ 0 \\ K_1 \end{pmatrix} \end{aligned}$$

— On ajoute K_2 à la dernière position :

$$X \leftarrow X + \begin{pmatrix} 0 \\ \vdots \\ 0 \\ K_2 \end{pmatrix} = A_1^2 \begin{pmatrix} 0 \\ \vdots \\ 0 \\ K_0 \end{pmatrix} + A_1 \begin{pmatrix} 0 \\ \vdots \\ 0 \\ K_1 \end{pmatrix} + \begin{pmatrix} 0 \\ \vdots \\ 0 \\ K_2 \end{pmatrix}$$
$$= \sum_{i=0}^{2} A^i \begin{pmatrix} 0 \\ \vdots \\ 0 \\ K_{2-i} \end{pmatrix}$$

La formule est donc vrai pour j de 0 à 2. On suppose que la formule est vrai au rang $n-1$. On a donc après l'étape $j = n-1$:

$$X = \sum_{i=0}^{n-1} A_1^i \begin{pmatrix} 0 \\ \vdots \\ 0 \\ K_{n-i-1} \end{pmatrix}$$

— L'étape $j = n$:
  — On met à jour le registre :

$$X \leftarrow A_1 X = A_1 \sum_{i=0}^{n-1} A_1^i \begin{pmatrix} 0 \\ \vdots \\ 0 \\ K_{n-i-1} \end{pmatrix} \quad = \sum_{i=0}^{n-1} A_1^{i+1} \begin{pmatrix} 0 \\ \vdots \\ 0 \\ K_{n-i-1} \end{pmatrix}$$

— On ajoute K_n à la dernière position :

$$X \leftarrow X + \begin{pmatrix} 0 \\ \vdots \\ 0 \\ K_n \end{pmatrix} = \sum_{i=0}^{n-1} A_1^{i+1} \begin{pmatrix} 0 \\ \vdots \\ 0 \\ K_{n-i-1} \end{pmatrix} + \begin{pmatrix} 0 \\ \vdots \\ 0 \\ K_n \end{pmatrix}$$

$$= A_1^1 \begin{pmatrix} 0 \\ \vdots \\ 0 \\ K_{n-1} \end{pmatrix} + A_1^2 \begin{pmatrix} 0 \\ \vdots \\ 0 \\ K_{n-2} \end{pmatrix} + \ldots + A_1^{n-1} \begin{pmatrix} 0 \\ \vdots \\ 0 \\ K_1 \end{pmatrix} + A_1^n \begin{pmatrix} 0 \\ \vdots \\ 0 \\ K_0 \end{pmatrix} + \begin{pmatrix} 0 \\ \vdots \\ 0 \\ K_n \end{pmatrix}$$

$$= A_1^0 \begin{pmatrix} 0 \\ \vdots \\ 0 \\ K_n \end{pmatrix} + A_1^1 \begin{pmatrix} 0 \\ \vdots \\ 0 \\ K_{n-1} \end{pmatrix} + A_1^2 \begin{pmatrix} 0 \\ \vdots \\ 0 \\ K_{n-2} \end{pmatrix} + \ldots + A_1^{n-1} \begin{pmatrix} 0 \\ \vdots \\ 0 \\ K_1 \end{pmatrix} + A_1^n \begin{pmatrix} 0 \\ \vdots \\ 0 \\ K_0 \end{pmatrix}$$

$$= \sum_{i=0}^{n} A_1^i \begin{pmatrix} 0 \\ \vdots \\ 0 \\ K_{n-i} \end{pmatrix}$$

Ainsi, on peut en déduire qu'après l'étape $j = 63$ on a :

$$X = \sum_{i=0}^{63} A_1^i \begin{pmatrix} 0 \\ \vdots \\ 0 \\ K_{63-i} \end{pmatrix}$$

Après la deuxième boucle On se place maintenant après la première boucle for. On va, comme précédemment, raisonner par récurrence.

On prend k pour identifier la k-ieme etape de la deuxieme boucle.

— L'étape $k = 0$

— On met à jour le registre :

$$Y \leftarrow A_1 X$$

— On ajoute IV_0 à la dernière position :

$$Y \leftarrow Y + \begin{pmatrix} 0 \\ \vdots \\ 0 \\ IV_0 \end{pmatrix} = A_1 X + \begin{pmatrix} 0 \\ \vdots \\ 0 \\ IV_0 \end{pmatrix}$$

— L'étape $k = 1$

— On met à jour le registre :

$$Y \leftarrow A_1 Y = A_1 \left[A_1 X + \begin{pmatrix} 0 \\ \vdots \\ 0 \\ IV_0 \end{pmatrix} \right] = A_1^2 X + A_1 \begin{pmatrix} 0 \\ \vdots \\ 0 \\ IV_0 \end{pmatrix}$$

— On ajoute IV_1 à la dernière position :

$$\begin{aligned} Y \leftarrow Y + \begin{pmatrix} 0 \\ \vdots \\ 0 \\ IV_1 \end{pmatrix} &= A_1^2 X + A_1 \begin{pmatrix} 0 \\ \vdots \\ 0 \\ IV_0 \end{pmatrix} + \begin{pmatrix} 0 \\ \vdots \\ 0 \\ IV_1 \end{pmatrix} \\ &= A_1^2 X + \sum_{i=0}^{1} A_1^i \begin{pmatrix} 0 \\ \vdots \\ 0 \\ IV_{1-i} \end{pmatrix} \end{aligned}$$

On a donc la formule vrai pour $k = 0, 1$. On suppose maintenant qu'elle est vrai après l'étape $n-1$:

— Pour $k = n-1$

$$Y = A_1^n X + \sum_{i=0}^{n-1} A_1^i \begin{pmatrix} 0 \\ \vdots \\ 0 \\ IV_{n-1-i} \end{pmatrix}$$

— L'étape $k = n$

— On met à jour le registre :

$$\begin{aligned} Y \leftarrow A_1 Y = A_1 \left[A_1^n X + \sum_{i=0}^{n-1} A_1^i \begin{pmatrix} 0 \\ \vdots \\ 0 \\ IV_{n-1-i} \end{pmatrix} \right] \\ = A_1^{n+1} X + \sum_{i=0}^{n-1} A_1^{i+1} \begin{pmatrix} 0 \\ \vdots \\ 0 \\ IV_{n-1-i} \end{pmatrix} \end{aligned}$$

— On ajoute IV_n à la dernière position :

$$\begin{aligned} Y \leftarrow Y + \begin{pmatrix} 0 \\ \vdots \\ 0 \\ IV_n \end{pmatrix} &= A_1^{n+1}X + \sum_{i=0}^{n-1} A_1^{i+1} \begin{pmatrix} 0 \\ \vdots \\ 0 \\ IV_{n-1-i} \end{pmatrix} + \begin{pmatrix} 0 \\ \vdots \\ 0 \\ IV_n \end{pmatrix} \\ &= A_1^{n+1}X + \sum_{i=0}^{n-1} A_1^{i+1} \begin{pmatrix} 0 \\ \vdots \\ 0 \\ IV_{n-1-i} \end{pmatrix} + A_1^0 \begin{pmatrix} 0 \\ \vdots \\ 0 \\ IV_n \end{pmatrix} \\ &= A_1^{n+1}X + \sum_{i=0}^{n} A_1^{i} \begin{pmatrix} 0 \\ \vdots \\ 0 \\ IV_{n-i} \end{pmatrix} \end{aligned}$$

On peut donc en déduire qu'après l'étape $k = 21$ on a :

$$Y = A_1^{22}X + \sum_{i=0}^{21} A_1^{i} \begin{pmatrix} 0 \\ \vdots \\ 0 \\ IV_{21-i} \end{pmatrix}$$

Étant donnée que le mécanisme est le même que pour celui de R_2, R_3 et R_4, on peut généraliser ces formules pour les autres registres.

1.10 Question 10

Apres les deux boucles, on modifie la 4^{ieme} valeur du registre $(R_{1,3})$. On a donc, a la fin de l'initialisation :

$$R1 = A_1^{22}X + \sum_{i=0}^{21} A_1^i \begin{pmatrix} 0 \\ \vdots \\ 0 \\ IV_{21-i} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

$$= A_1^{22} \left[\sum_{i=0}^{63} A_1^i \begin{pmatrix} 0 \\ \vdots \\ 0 \\ K_{63-i} \end{pmatrix} \right] + \sum_{i=0}^{21} A_1^i \begin{pmatrix} 0 \\ \vdots \\ 0 \\ IV_{21-i} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Or, VI est définit par le vecteur nul.

$$R1 = A_1^{22} \left[\sum_{i=0}^{63} A_1^i \begin{pmatrix} 0 \\ \vdots \\ 0 \\ K_{63-i} \end{pmatrix} \right] + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

voir sage

1.11 Question 11

Avec les formules que l'on a obtenue à la question 9, ont peut en déduire une formule sur les registres en fonction du vecteur d'initialisation. On note $R_1(z_0)$, $R_1(z_1)$, $R_1(z_2)$ les registres $R1$ ayant produit les suites z_0, z_1, z_2 respectivement.

— Pour $R_1(z0)$ on a $IV = (0, \ldots, 0)$

$$\begin{aligned} R_1(z_0) &= A_1^{22}X + \sum_{i=0}^{21} A_1^i \begin{pmatrix} 0 \\ \vdots \\ 0 \\ IV_{21-i} \end{pmatrix} \\ &= A_1^{22}X + \sum_{i=0}^{21} A_1^i \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix} \\ &= A_1^{22}X \end{aligned}$$

— Pour $R_1(z_1)$ on a $IV = (0, \ldots, 0, 1)$

$$\begin{aligned} R_1(z_1) &= A_1^{22}X + \sum_{i=0}^{21} A_1^i \begin{pmatrix} 0 \\ \vdots \\ 0 \\ IV_{21-i} \end{pmatrix} \\ &= A_1^{22}X + \left[A_1^0 \begin{pmatrix} 0 \\ \vdots \\ 0 \\ IV_{21} \end{pmatrix} + A_1^1 \begin{pmatrix} 0 \\ \vdots \\ 0 \\ IV_{20} \end{pmatrix} + \ldots + A_1^{21} \begin{pmatrix} 0 \\ \vdots \\ 0 \\ IV_0 \end{pmatrix} \right] \\ &= A_1^{22}X + \left[A_1^0 \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} + A_1^1 \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix} + \ldots + A_1^{21} \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix} \right] \\ &= A_1^{22}X + A_1^0 \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} \\ &= R_1(z_0) + A_1^0 \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} \end{aligned}$$

— Pour $R_1(z_2)$ on a $IV = (0, \ldots, 1, 0)$

$$
\begin{aligned}
R_1(z_2) &= A_1^{22}X + \sum_{i=0}^{21} A_1^i \begin{pmatrix} 0 \\ \vdots \\ 0 \\ IV_{21-i} \end{pmatrix} \\
&= A_1^{22}X + \left[A_1^0 \begin{pmatrix} 0 \\ \vdots \\ 0 \\ IV_{21} \end{pmatrix} + A_1^1 \begin{pmatrix} 0 \\ \vdots \\ 0 \\ IV_{20} \end{pmatrix} + \ldots + A_1^{21} \begin{pmatrix} 0 \\ \vdots \\ 0 \\ IV_0 \end{pmatrix} \right] \\
&= A_1^{22}X + \left[A_1^0 \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix} + A_1^1 \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} + \ldots + A_1^{21} \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix} \right] \\
&= A_1^{22}X + A_1^1 \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} \\
&= R_1(z_0) + A_1^1 \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}
\end{aligned}
$$

1.12 Question 12

Le problème que nous avions pour effectuer une attaque sur 228 bits était que le nombre d'équation en les variables des registres n'était pas suffisant. Effectivement, nous avions 228 équations pour 655 monômes à déterminer.

Mais nous avons maintenant la possibilité d'exprimer les autres registres ayant produit z_1 et z_2 à partir des registres ayant produits z_0. On va donc pourvoir accumuler les équations dans la matrice de linéarisation.

1.12.1 Les registres

Avant de générer les équations de la suite chiffrante, on va définir les registres comme suit :

— Pour générer les équations de z_0 :
  — $R_1(z_0) = (x_0, \ldots, x_{18})$
  — $R_2(z_0) = (x_{19}, \ldots, x_{40})$
  — $R_3(z_0) = (x_{41}, \ldots, x_{63})$
— Pour générer les équations de z_1 :

- $R_1(z_0) = (x_0, \ldots, x_{18}) + (0, \ldots, 0, 1)$
- $R_2(z_0) = (x_{19}, \ldots, x_{40}) + (0, \ldots, 0, 1)$
- $R_3(z_0) = (x_{41}, \ldots, x_{63}) + (0, \ldots, 0, 1)$

— Pour générer les équations de z_2 :

- $R_1(z_0) = (x_0, \ldots, x_{18}) + A_1 vect(0, \ldots, 0, 1)$
- $R_2(z_0) = (x_{19}, \ldots, x_{40}) + A_2 vect(0, \ldots, 0, 1)$
- $R_3(z_0) = (x_{41}, \ldots, x_{63}) + A_3 vect(0, \ldots, 0, 1)$

1.12.2 Linéarisation

Pour obtenir la linéarisation, on va appliquer le même algorithme utilisé précédemment prenant en entré la liste des équations et qui nous renvoie une matrice dont les colonnes correspond aux monômes possibles et les lignes les équations, et le vecteur qui contient un 1 si celui ci apparaît dans l'équation de la même ligne.

La liste d'équation a rentrer sera les listes des équations concaténées : $[z_0] + [z_1] + [z_2]$

Nous allons donc obtenir $3 * 228 = 684$ équations, une valeur supérieur à 655, ce qui nous permettra d'appliquer, comme dans la question 8, la résolution d'équation.

1.12.3 La clé K

Une fois la solution obtenue, on en déduit les valeurs des registres R_1, R_2 et R_3 ayant produit z_0.

Maintenant que nous connaissons les registres, nous allons procéder de la même façon pour retrouver K, c'est à dire définir K à partir de variables x_i puis initialiser les registres afin d'obtenir des équations linéaires en les x_i.

1.13 Question 13

Voir code sage.

1.14 Question 14

On pourrait imaginer une attaque sur R_4 par la méthode de force brute. On aurait dans ce cas 2^{17} cas possibles.

Pour que l'attaque fonctionne, il faudrait, à chaque registre R_4, non seulement calculer les registres, mais en plus les vérifier en générant la suite chiffrante. En faisant le test sur sage, on trouve un temps de 1.3 secondes. L'attaque durerait en moyenne $2^{17} \times 1.3$ secondes, ce qui nous donne 1j 23h 19min 53sec.

Un temps qui paraît raisonnable, surtout en sachant que le code n'est pas optimisé.

1.15 Question 15

Pour eviter cette attaque, il faudrait augmenter la taille du registre R_4.

1.15.1 complexité de l'attaque

Pour voir cela de façon plus formelle, on va grossierement etudier la complexité de l'attaque.

— Calcul des équations des registres :
Pour définir les équations des registre on va principalement s'intéresser à la complexité de la fonction `a5_2_exec()`. Effectivement, on procède comme suit :
  — On associe les variables aux registres
  — On applique un calcul sur les registres si c'est nécessaire
  — On génère la suite chiffrante
  On peut facilement en déduire que celle ci est en temps constant dans la première boucle, puis dépend de N dans la deuxième. Ainsi, la complexité est en $O(N)$.

— Linearisation :
Le plus important dans cette partie est la complexité de la fonction `create_matrix()` car c'est elle qui va créer la matrice de taille $N \times l_M$ avec l_M le nombre de monômes possibles. La complexité de cette fonction est donc en $O(N \times l_M)$.

— Calcul de la solution :
Après quelques recherches il s'avère que la résolution d'équation linéaire par cette méthode est en $O(n^3)$ avec n le nombre de valeurs à déterminer, soit, dans notre cas, l_M valeurs. La complexité est donc en $O(l_M^3)$

— Récupération :
On va ici s'intéresser à la fonction `get_value_sol()` qui va parcourir la liste des indices précalculée afin de recuperer les valeurs à ces meme indices. La complexité de cette fonction est en $O(t_K)$ avec t_k la taille de la clé K.

— Test :
A cette etape on teste si les registres nous donne bien la bonne valeur de la suite chiffrante. C'est donc aussi la complexité de `a5_2_exec()` qui est en $O(N)$

Et si on augmente la taille de la clée ? On a eu le cas $t_R1 + t_R2 + t_R3 = t_K$ qui nous arrange car il suffisait de retouver les registres pour ensuite creer des equations. Mais si on avait eu $t_R1 + t_R2 + t_R3 < t_K$ l'attaque aurait ete impossible.

2 Cryptanalyse de RSA

2.1 Question 16

Création de la fonction :

1. `gen_keys(k,delta)`

Voir code sage pour les commentaires.

2.2 Question 17

2.2.1 Montrons que (x_0, y_0) est racine de $f(x, y)$

Démonstration. Pour commencer, on a :

$$\begin{aligned}\varphi(n) &= (p-1)(q-1)\\ &= pq - p - q + 1\\ &= N - (p+q) + 1\\ &= N + 1 - (p+q)\end{aligned}$$

On a aussi :

$$eq = 1 + x_0\varphi(n) \iff 1 + x_0\varphi(n) \equiv 0 \mod e$$

Ce qui nous donne :

$$\begin{aligned}0 &\equiv 1 + x_0\varphi(n) \mod e\\ &\equiv 1 + x_0(N + 1 - (p+q)) \mod e\\ &\equiv 1 + x_0(N + 1 + y_0) \mod e\\ &\equiv 1 + x_0(A + y_0) \mod e\\ 0 &\equiv f(x_0, y_0)\end{aligned}$$

Ainsi, (x_0, y_0) est racine de $f(x, y)$ □

2.2.2 Montrons que $x_0 \approx e^{\delta}$ et $|y_0| \approx e^{1/2}$

Démonstration.

On suppose que $e \approx \varphi(n) \approx N$

— Montrons que $x_0 \approx e^{\delta}$

$$\begin{aligned}1 + x_0\varphi(n) &= ed\\ x_0\varphi(n) &\approx ed\end{aligned}$$

On a $\varphi(n) \approx e$

$$\begin{aligned}x_0 e &\approx ed\\ x_0 &\approx d\end{aligned}$$

On a $d \approx N^\delta$

$$x_0 \approx N^\delta$$

On a $N \approx e$

$$x_0 \approx e^\delta$$

— Montrons que $|y_0| \approx e^{1/2}$

$$y_0 = -(p+q)$$

On sait que : $p \approx 2^k \approx q$

$$y_0 \approx -2^{k+1} \approx -2^k$$

On sait que $N \approx 2^{2k}$

$$y_0^2 \approx (-2^k)^2 \approx 2^{2k} \approx N \approx e$$

On a donc :

$$y_0^2 \approx e$$
$$|y_0| \approx e^{1/2}$$

□

2.3 Question 18

2.3.1 Montrons que $\bar{g}_{i,k}(u_0, x_0) \equiv 0 \mod e^m$

Démonstration. On va développer $\bar{g}_{i,k}(u_0, x_0)$ pour observer le résultat modulo e^m. On a donc :

$$\begin{aligned}\bar{g}_{i,k}(u_0, x_0) &= x_0^i \bar{f}(u_0, x_0)^k e^{m-k} \\ &= x_0^i f(x_0, y_0)^k e^{m-k}\end{aligned}$$

Or, on a vu précédemment que

$$f(x_0, y_0) \equiv 0 \mod e$$

ce qui équivaut à dire qu'il existe $v \in \mathbb{Z}$ tel que $f(x_0, y_0) = ev$. Revenons donc à notre formule :

$$\begin{aligned}\bar{g}_{i,k}(u_0, x_0) &= x_0^i f(x_0, y_0)^k e^{m-k} \\ &= x_0^i (ev)^k e^{m-k} \\ &= x_0^i v^k e^k e^{m-k} \\ &= x_0^i v^k e^{m-k+k} \\ &= x_0^i v^k e^m \equiv 0 \mod e^m\end{aligned}$$

Le couple (u_0, y_0) est donc bien une racine de tout polynome $\bar{g}_{i,k}(u, x)$ modulo e^m. □

2.3.2 Montrons que $\bar{h}_{j,k}(u_0, x_0, y_0) \equiv 0 \mod e^m$

De même, on va développer la fonction et observer le résultat.

$$\begin{aligned}
\bar{h}_{j,k}(u_0, x_0, y_0) &= y_0^j \bar{f}(u_0, x_0)^k e^{m-k} \\
&= y_0^j f(x_0, y_0)^k e^{m-k} \\
&= y_0^j (ev)^k e^{m-k} \\
&= y_0^j v^k e^k e^{m-k} \\
&= y_0^j v^k e^{m-k+k} \\
&= y_0^j v^k e^m \equiv 0 \mod e^m
\end{aligned}$$

On a bien (x_0, y_0), une racine de tout polynome $\bar{h}_{j,k}(u_0, x_0, y_0)$.

2.4 Question 19

Le but ici est de montrer qu'il y a en tout, l polynômes possibles.

2.4.1 Dénombrement de M

On va commencer par dénombrer l'ensemble M.

Première partie : On pose

$$M_1 := \{x^{k-i}u^i,\ \forall k \in \{0, \ldots, m\}, \forall i \in \{0, \ldots, k\}\}$$

Pour dénombrer le nombre de valeurs dans cette ensemble, on va développer les étapes.

- pour $k = 0$, $i = \{0\}$, soit 1 valeur
- $k = 1$, $i = \{0, 1\}$, soit 2 valeurs
- $k = 2$, $i = \{0, 1, 2\}$, soit 3 valeurs

  ⋮
- $k = m$, $i = \{0, 1, 2, \ldots, m\}$, soit $m + 1$ valeurs

Ce qui nous donne comme nombre de valeurs : $1 + 2 + 3 + \ldots + (m + 1)$ Ainsi, on a :

$$card(M_1) = \sum_{i=0}^{m} (i + 1)$$

Deuxième partie : On pose :

$$M_2 := \{y^j u^k,\ \forall j \in \{1, \ldots, t\}, \forall k \in \{\lfloor m/t \rfloor j, \ldots, m\}\}$$

On va procéder de la même façon :

— pour $j = 1$, $k = \{\lfloor m/t \rfloor, \ldots, m\}$, soit $m - \lfloor m/t \rfloor + 1$ valeurs
— $j = 2$, $k = \{\lfloor m/t \rfloor \times 2, \ldots, m\}$, soit $m - \lfloor m/t \rfloor \times 2 + 1$ valeurs
⋮
— $j = t$, $k = \{\lfloor m/t \rfloor \times t, \ldots, m\}$, soit $m - \lfloor m/t \rfloor \times t + 1$ valeurs

Ainsi, on a :

$$card(M_2) = \sum_{i=1}^{t} (m - \lfloor m/t \rfloor \times i + 1)$$

Intersection : Pour dénombrer correctement, il faudrait aussi compter le nombre d'élément dans l'intersection. On va donc chercher des éléments de tels que

$$x^{k-i}u^i = y^j u^k$$

Pour que ça soit possible, on commence par éliminer les x et y, or, pour ça, il faudrait que l'on ai $j = 0$ ce qui n'est pas possible. L'intersection est donc vide.

On peut donc en déduire que :

$$l = \sum_{i=0}^{m} (i + 1) + \sum_{i=1}^{t} (m - \lfloor m/t \rfloor \times i + 1)$$

2.4.2 Dénombrement des polynômes

On va maintenant dénombrer les polynômes possible en fonction de leurs indices.

Famille de $\bar{g}_{i,k}(u, x)$ On va noter $F_{\bar{g}}$ la famille de $\bar{g}_{i,k}(u, x)$ pour $k \in \{0, \ldots, m\}$ et $i \in \{0, \ldots, m - k\}$.
— pour $k = 0$, $i = \{0, \ldots, m\}$, soit $m + 1$ valeurs
— $k = 1$, $i = \{0, \ldots, m - 1\}$, soit m valeurs
⋮
— $k = m$, $i = \{0, \ldots, m - m\}$, soit 1 valeur

On retrouve :

$$card(F_{\bar{g}}) = \sum_{i=0}^{m} (m - i + 1) = \sum_{i=0}^{m} (i + 1)$$

Famille de $\bar{h}_{j,k}(u, x, y)$ On va noter $F_{\bar{h}}$ la famille de $\bar{h}_{j,k}(u, x, y)$ pour $j \in \{1, \ldots, t\}$ et $k \in \{\lfloor m/t \rfloor j, \ldots, m\}$. La, on retrouve exactement les même indices que pour le calcul du cardinal de $M2$, ainsi on peut directement en déduire le nombre de polynômes possibles.

On retrouve donc :

$$card(F_{\bar{h}}) = \sum_{i=1}^{t} (m - \lfloor m/t \rfloor \times i + 1)$$

Ainsi, on retrouve bien $l = card(F_{\bar{g}}) + card(F_{\bar{h}})$

2.5 Question 20

On va développer les polynômes pour montrer que les monômes appartiennes bien à l'ensemble M.

2.5.1 Développement de $\bar{g}_{i,k}(u,x)$

$$\begin{aligned}
\bar{g}_{i,k}(u,x) &= x^i \bar{f}(u,x)^k e^{m-k} \\
&= x^i (u + Ax)^k e^{m-k} \\
&= x^i \left[\sum_{\alpha=0}^{k} \binom{k}{\alpha} u^\alpha (Ax)^{k-\alpha} \right] e^{m-k} \\
&= \sum_{\alpha=0}^{k} \binom{k}{\alpha} x^i x^{k-\alpha} u^\alpha A^{k-\alpha} e^{m-k} \\
&= \sum_{\alpha=0}^{k} \binom{k}{\alpha} x^{i+k-\alpha} u^\alpha A^{k-\alpha} e^{m-k}
\end{aligned}$$

On trouve le monôme $x^{i+k-\alpha}u^\alpha$ avec $\alpha \in \{0,\dots,k\}$, $k \in \{0,\dots,m\}$ et $i \in \{0,\dots,m-k\}$. Pour montrer que ce monôme se trouve bien dans M, il faut montrer que les puissances sont dans le bon intervalle.

- La puissance de u doit être dans $[0,\dots,k]$, ce qui est la cas de α.
- Le puissance de x doit être dans $[0,\dots,m]$, il faut donc montrer que c'est le cas pour $i+k$.
  - Si $k = 0$, on a $i = m$
  - Si $k = m$, on a $i = 0$

  On a bien $0 \leq i + k \leq m$.

Ainsi, les monômes de $\bar{g}_{i,k}(u,x)$ sont bien dans M.

2.5.2 Développement de $\bar{h}_{j,k}(u,x,y)$

$$\begin{aligned}
\bar{h}_{j,k}(u,x,y) &= y^j \bar{f}(u,x)^k e^{m-k} \\
&= y^j (u + Ax)^k e^{m-k} \\
&= y^j \left[\sum_{\alpha=0}^{k} \binom{k}{\alpha} u^\alpha (Ax)^{k-\alpha} \right] e^{m-k} \\
&= \sum_{\alpha=0}^{k} \binom{k}{\alpha} y^j x^{k-\alpha} u^\alpha A^{k-\alpha} e^{m-k}
\end{aligned}$$

On trouve le monôme $y^j u^\alpha x^{k-\alpha}$ avec $\alpha \in \{0, \ldots, k\}$, $j \in \{1, \ldots, t\}$ et $k \in \{\lfloor m/t \rfloor j, \ldots, m\}$. De même que précédemment, on va étudier les puissances.

- Regardons le cas trivial ou $x = 1$ soit, le cas où $k - \alpha = 0$.
  Si $k - \alpha = 0$ alors $k = \alpha$. On a donc $\alpha \in \{\lfloor m/t \rfloor j, \ldots, m\}$.
- Si on a $j > k - \alpha$ alors on va pouvoir remplacer $(xy)^{k-\alpha}$ par $(u-1)^k - \alpha$, il ne restera plus de x, et donc ce monome sera bien dans M_2
- Si, à l'inverse, le degré de x est supérieure ou égal à celui de y, alors dans ce cas on va se retrouver avec un monôme appartenant à M_1.

2.6 Question 21

Création des fonctions :

1. `change_xy(P)`
2. `f_bar(A,u,x)`
3. `g_bar(A,e,m,i,k,u,x)`
4. `h_bar(A,e,m,t,j,k,u,x,y)`
5. `g_bar_familly(A,e,m,u,x)`
6. `h_bar_familly(A,e,m,t,u,x,y)`
7. `find_monom_list(m,t)`
8. `change_xy_list(H)`
9. `create_dico(M)`
10. `evaluate(x,y,u,list_poly)`
11. `create_basis_matrix(A,e,m,t,U,X,Y,monom_list,dico)`

Voir code sage pour les commentaires.

2.7 Question 22

Création de la fonction :

1. `change_u(P)`

Voir code sage pour les commentaires.

2.8 Question 23

On souhaite montrer $P(x_0, y_0) = 0$ dans $\mathbb{Z}$.

Démonstration : Posons [1]

$$P(x_0, y_0) = \sum_{i=1}^{\omega} a_i x^{\alpha_i} y^{\beta_i} \text{ avec } 0 \leq \alpha_i \leq \omega \text{ et } 0 \leq \beta_i \leq \omega$$

1. On par de $i = 1$ car on a au plus ω valeurs possibles.

On peut donc aussi poser :

$$|P(x_0, y_0)| = \left|\sum_{i=1}^{\omega} a_i x_0^{\alpha_i} y_0^{\beta_i}\right| \leq \sum_{i=1}^{\omega} \left|a_i x_0^{\alpha_i} y_0^{\beta_i}\right| \leq \sum_{i=1}^{\omega} |a_i| X^{\alpha_i} Y^{\beta_i}$$

Grâce à l'inégalité de Cauchy-Schwarz et au fait que $|x_0| \leq X$ et $|y_0| \leq Y$.

On peut écrire notre somme sous la forme d'un produit scalaire :

$$\sum_{i=1}^{\omega} |a_i| X^{\alpha_i} Y^{\beta_i} = \langle (|a_1| X^{\alpha_1} Y^{\beta_1}, \ldots, |a_\omega| X^{\alpha_\omega} Y^{\beta_\omega}), (1, \ldots, 1) \rangle \quad (1)$$

$$= ||P(xX, yY)|| \times ||(1, \ldots, 1)|| \quad (2)$$

$$= ||P(xX, yY)|| \times \sqrt{\omega} < e^m \quad (3)$$

Avant tout cela, on a supposé que $P(x_0, y_0) \equiv 0 \ mod \ e^m$
i.e., il existe $Q \in \mathbb{Z}$ tel que $P(x_0, y_0) = Q \times e^m$. Or, on vient de voir que $|P(x_0, y_0)| < e^m$, donc $Q = 0$, ce qui nous donne bien :

$$P(x_0, y_0) = 0 \text{ dans } \mathbb{Z}$$

2.9 Question 24

Voir code sage pour la démarche à suivre.

2.9.1 Trouver $P1$ et $P2$:

Le but de cette algorithme est de trouver deux polynômes qui respectent certaines propriété afin d'en déduire leurs racines communes. Voici donc l'algorithme à suivre :

- récupérer et convertir un vecteur de la base sous forme de polynôme P_1
- retirer les bornes X, Y et U des coefficients
- substituer les variables u par $1 + xy$
- si $||P_1(xX, yY)|| < \frac{e^m}{\sqrt{\omega}}$
  - récupérer un autre polynôme P_2
  - retirer les bornes X, Y et U des coefficients
  - substituer les variables u par $1 + xy$
  - si $||P_2(xX, yY)|| < \frac{e^m}{\sqrt{\omega}}$
    - calculer le resultant en retirant la variable y
    - si celui ci à au moins deux racines (0 et x_0)
      - retourner x_0

A la dernière étape, on est sûr d'avoir trouvé de bons polynômes P_1 et P_2.

2.9.2 Trouver x_0 et y_0 :

Finalement, on obtient la fonction principale `get_2_good_poly(BasisLLL,idico,e,m)` qui, en fonction de la base, du dictionnaire inverse, de e et m, renvoie $P_1, P_2, Sol, racines$ avec :

— P_1 et P_2 deux polynômes qui respectent les conditions.
— Sol le résultant de P_1 et P_2 en fonction de y.
— $racines$ les racines du polynôme Sol

La demarche à suivre est maintenant la suivante :
— on à x_0 comme la racine non nul du resultant ;
— on evalue $P_1(x_0, y)$;
— on trouve y_0 comme la racine de ce polynome ;

Finalement, on obtient $y_0 = -(p+q)$. Or, on a :

$$\begin{aligned}\varphi(N) &= (p-1)(q-1)\\ &= pq - p - q + 1\\ &= N + 1 - (p+q)\\ &= A + y_0\end{aligned}$$

Ainsi, on a $\varphi(N) = A + y_0$.

2.9.3 Retrouver d

On a la proposition suivante :

Proposition 2.1. *Connaître n et $\varphi(n)$ est équivalent à connaître p et q.*

Démonstration. Si p et q sont deux nombres premiers, alors,

$$\begin{aligned}\varphi(n) &= (p-1)(q-1)\\ &= pq - p - q + 1\\ &= n - p - q + 1\\ 0 &= n - p - q + 1 - \varphi(n)\\ &= pn - p^2 - pq + p - p\varphi(n)\\ &= -p^2 + pn - n + p - p\varphi(n)\\ &= -p^2 + p(n + 1 - \varphi(n)) - n\\ 0 &= p^2 - p(n + 1 - \varphi(n)) + n\end{aligned}$$

Ainsi, nous avons un polynôme dont p est racine. On peut donc calculer le

discriminant :

$$\begin{aligned}\Delta &= b^2 - 4ac \\ &= (n + 1 - \varphi(n))^2 - 4n\end{aligned}$$

On a pour finir deux solutions :

$$\frac{-b - \sqrt{\Delta}}{2a} = p$$
$$\frac{-b + \sqrt{\Delta}}{2a} = q$$

Si $p < q$.

□

On peut en déduire la formule suivante :

— Pour
  — $a = 1$
  — $b = -(N + 1 - \varphi(N))$
  — $c = N$

$$\Delta = b^2 - 4ac$$
$$d = \frac{-b - \sqrt{\Delta}}{2a}$$

On pourra donc renvoyer $PGCD(d, N)$

2.9.4 Observations :

On remarque, après plusieurs tests [2] effectués, que lorsque l'attaque fonctionne, le polynômes P_1 est toujours le premier vecteur de la base de la matrice. On pourrait donc en déduire que l'on pourrait se contenter du premier vecteur pour P_1.

On remarque aussi que, avant de trouver $P2$, on parcourt environ $\frac{\omega}{2}$ vecteurs de la base.

2.10 Attaque sur le fichier RSA.sage

Je ne suis pas parvenu à factoriser N en ayant exécuté l'algorithme avec les valeurs suivantes :

— m : de 2 à 11, et 12 arrêté à mi-chemin (25% de la recherche des polynômes).
— δ : de 0.20 à 0.27

2. L'expérience a été effectuée avec 100 produits de 2 premiers de 1024 bits et comme paramètres $\delta = 0.2$, $m = 2$ et $t = 1$