Défi c0d1ngUP : Fabrique de spores

c0d1ngUP team

Le nouveau défi que nous vous proposons provient de l'édition 2024 de c0d1ngUP et s'intitule *Fabrique de spores*. Le thème de l'édition était X-Files et vous accompagnerez donc Dana Scully. Cette fois encore, le texte du défi est exactement celui qui a été proposé lors du challenge c0d1ngUP 2024 (d'où la présence de quelques précisions qu'il ne serait pas nécessaire d'apporter aux lecteurs de 1024).

Par ailleurs, nous fournissons comme promis des pistes de résolution pour le défi proposé dans le numéro 24 : *Chiffrement Cybermen Daleks*.

Nouveau défi — *Fabrique de spores*

Suite à l'aventure hallucinogène avec un fongus[1] géant, Scully a décidé d'étudier plus en détail le champignon. Il fabrique des spores continuellement selon un processus particulièrement ordonné, avant de les libérer par salves. Les spores sont organisées sur un réseau cubique. Chaque nœud de ce réseau contient une spore, d'un type ou d'un autre. Il y a 4 types de spores, qu'on appellera spores de type 0, 1, 2 ou 3.

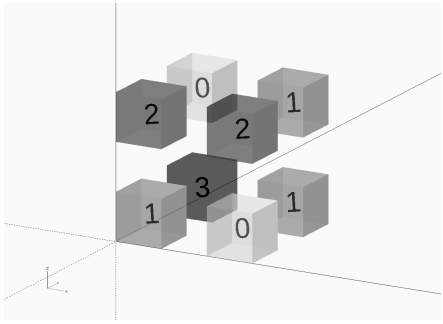

Fig. 1. Les 8 spores initiales.

Toutes les figures qui suivent — avec une représentation en couleur des spores (type 0 en magenta, type 1 en rouge, type 2 en vert, type 3 en bleu) — sont disponibles en ligne dans ce document[2].

1. https://fr.wikipedia.org/wiki/Fungi.
2. https://codingup.socinfo.fr/1024/25/pb/figures.pdf.

DOI:10.48556/SIF.1024.25.203

Initialement, dans le champignon, le réseau de spores n'a que 8 nœuds organisés en cube, portant 8 spores (figure 1).

À intervalles réguliers, le réseau de spores se transforme. Les nœuds semblent s'écarter, donnant naissance à de nouveaux nœuds (dans les trois vues, sous des angles différents, figures 2 à 4, on voit qu'il y a 19 nouveaux nœuds créés).

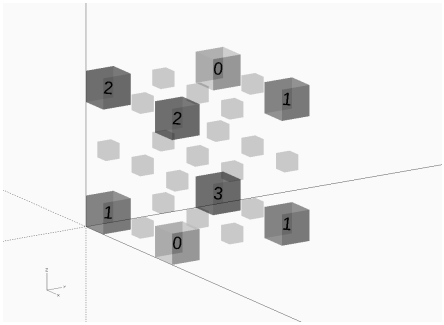

Fig. 2. Emplacement pour les nouvelles spores.

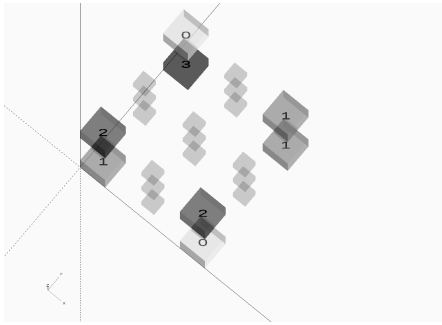

Fig. 3. Emplacement pour les nouvelles spores.

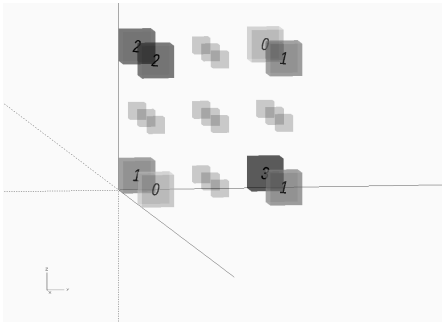

Fig. 4. Emplacement pour les nouvelles spores.

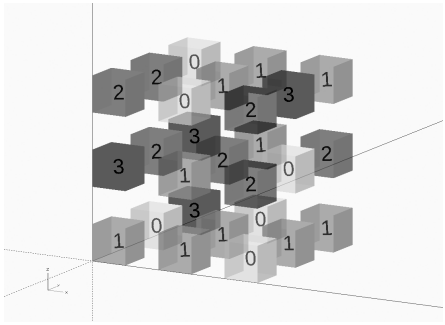

Fig. 5. Génération suivante : 27 spores.

Puis, chaque nouveau nœud créé se retrouve occupé par une nouvelle spore dont le type (0, 1, 2 ou 3) dépend des nœuds voisins (figure 5). Une représentation éclatée est visible à la figure 6.

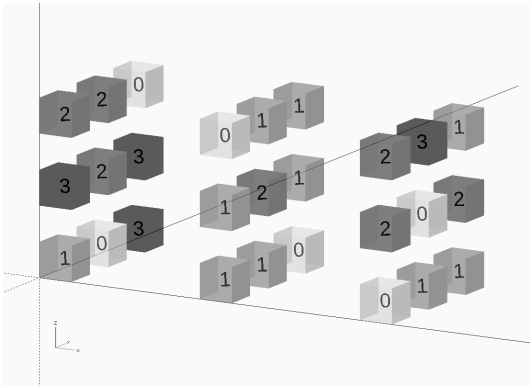

Fig. 6. Vue éclatée des 27 spores.

Un nœud a potentiellement 26 voisins (songez à un Rubik's cube, il y a 26 cubes autour du cube central, qu'on ne voit pas). Les règles d'expansion du réseau de spores font qu'un nouveau nœud aura un nombre de voisins contenant des spores égal à 2, 4 ou 8. Dans l'exemple qui précède, il y a 12 nœuds ayant 2 spores voisines, 6 nœuds ayant 4 spores voisines, et 1 seul ayant 8 spores voisines.

Dans tous les cas, Scully a pu déterminer les règles d'apparition des nouvelles spores sur les nouveaux nœuds. Ces règles varient selon le type de spores sur les noœuds voisins : ce nouveau nœud accueillera une spore de type k, où k est la somme des types des nœuds voisins modulo 4 (k est donc bien un nombre entre 0 et 3). Comment les spores font-elles ce calcul ? On l'ignore encore. Vérifions cependant ce que cela donne sur quelques exemples.

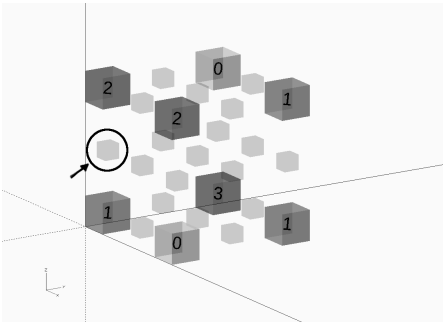

Fig. 7. Quelle spore occupera cette place ?

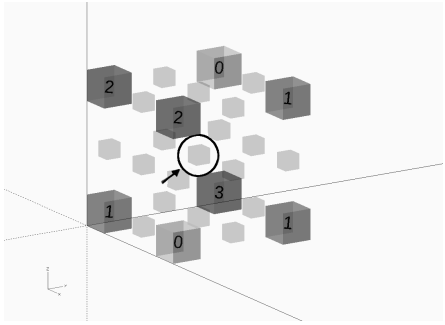

Fig. 8. Quelle spore occupera le nœud central ?

Sur la figure 7, voyons quelle spore occupe le nœud pointé par une flèche. Il n'y a que deux spores voisines, de type 1 et 2 (rouge et vert). La somme modulo 4 donne 3, et le nœud sera donc de type 3 (bleu).

Traitons un second exemple avec la spore portée par le nœud central (figure 8).

Ce nœud avait initialement 8 spores voisines. La somme de leur type vaut : 2 (vert) + 1 (rouge) + 1 (rouge) + 0 (magenta) + 2 (vert) + 0 (magenta) + 3 (bleu) + 1 (rouge) = 10. Le reste modulo 4 est 2, la cellule centrale sera donc de type 2 (vert).

Lors de l'étape suivante, la transformation s'opère de nouveau. Les nœuds s'écartent, donnant naissance à de nouveaux nœuds (figure 9).

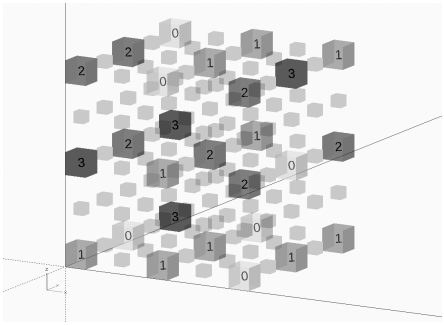

Fig. 9. Emplacement pour les nouvelles spores.

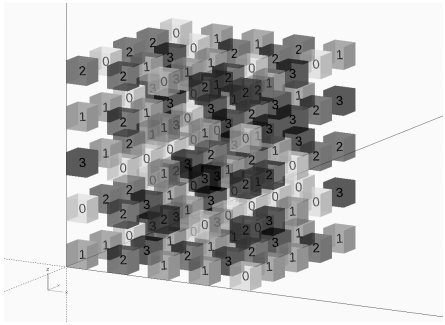

Fig. 10. Apparition des nouvelles spores.

Les apparitions de spores sont synchrones : toutes les spores des nouveaux nœuds apparaissent en même temps. Ce que Scully a pu observer à l'étape suivante est visible sur la figure 10.

Pour plus de lisibilité, 5 couches de spores en vue éclatée sont visibles à la figure 11. Lorsque la réserve de spores est trop grosse, les spores sont lâchées et l'effet du nuage qui se dégage dépend très sensiblement du nombre de spores de chaque type, selon une alchimie qui échappe actuellement à l'entendement.

Scully essaie donc de déterminer, à chaque étape, combien il y a de spores de chaque type, en consignant simplement 4 nombres indiquant le nombre de spores de type 0, 1, 2 et 3 :

– elle a relevé initialement : 2, 3, 2, 1 (on rappelle que l'étape initiale est toujours la même, avec 8 spores toujours à la même place comme indiqué sur le premier schéma) ;
– puis après une étape : 6, 10, 7, 4 ;
– après 2 étapes : 30, 36, 31, 28.

Ensuite, il devient difficile de comptabiliser les spores au microscope. Votre travail est de réaliser la simulation informatique qui pourra prédire le nombre exact de chaque type de spores. Les premiers essais sont concluants, et ont été vérifiés par un expert :
– après 3 étapes : 196, 192, 155, 186 ;
– après 4 étapes : 1162, 1276, 1207, 1268.

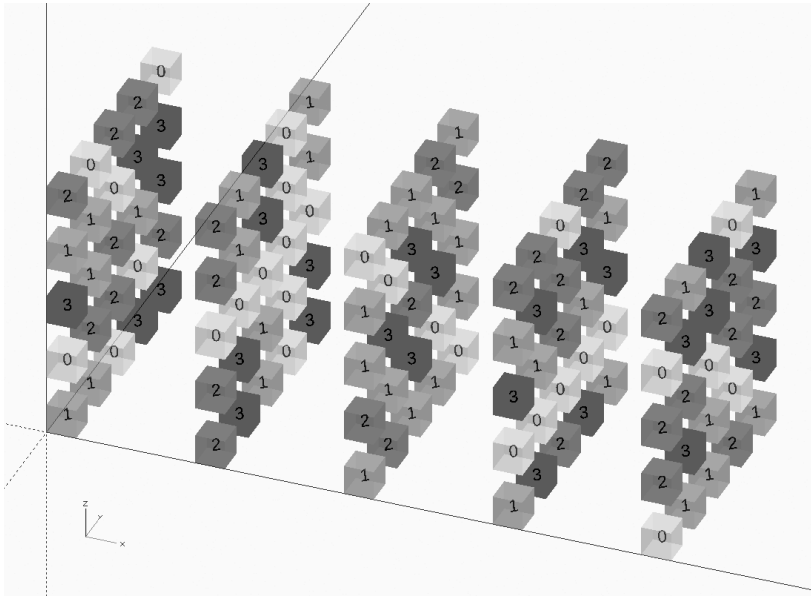

Fig. 11. Vue éclatée à l'étape 3.

Les spores sont lâchées de leur réserve après la 17^{e} étape. Validez le défi en indiquant le nombre de spores de chaque type.

Solution du défi précédent
Chiffrement Cybermen Daleks

On dispose d'un code en C réalisant un chiffrement, ainsi qu'un texte en clair et le texte chiffré correspondant. Le défi consiste à décrypter un autre message chiffré avec la même clé. Voici les données initiales :

– Code C de la méthode de chiffrement[3] ;
– texte en clair connu : « `EXTERMINER` » ;
– résultat du chiffrement du message précédent : «`MA.AYTBUAY` » ;
– message à décrypter[4] : « `:RMOP,DOCC,QTUY,QMHWVA␣L␣DCUUGYMHN␣QUQXWK!KJGL?FRD␣EL,OJQMOXUIYSDOBMQF` ».

Dans la méthode de chiffrement proposée (voir le code ci-dessus), seuls les caractères ASCII majuscules, et quelques symboles de ponctuation sont utilisés. À un tel caractère, on associe exactement un entier entre 0 et 31 (fonction `letter_num`).

La méthode consiste ainsi à chiffrer une séquence d'entiers entre 0 et 31, en appliquant deux chiffrements par blocs successifs. La taille des blocs, dans les deux cas, est de 5 caractères (ou entiers).

Le premier chiffrement utilise une clé k_1 de 5 caractères (donc 5 entiers entre 0 et 31), et réalise un « ou exclusif » avec le bloc en clair. Le résultat est donc aussi une séquence de 5 entiers entre 0 et 31.

La seconde méthode combine chaque entier d'un bloc, pour construire un nombre qui s'écrit sur 25 bits (5 entiers de 5 bits), c'est à dire un nombre compris entre 0 et $33\,554\,431 = 2^{25} - 1$. Cet entier est multiplié par la seconde clé numérique k_2 (entier inférieur à 2^{25}), puis le résultat, modulo 2^25 est redécoupé en 5 entiers de 5 bits pour donner le bloc chiffré.

Le chiffrement est réversible (ce qui est nécessaire) si et seulement si k_2 est inversible modulo 2^{25} (ce qui est encore équivalent[5].] au fait que k_2 est premier avec 2^{25}). Chaque bloc est chiffré indépendamment des autres (mode de chiffrement ECB, pour *electronic code book*).

Puisque l'on connaît un message en clair et le message chiffré correpondant, il est possible de tester tous les couples de clés, jusqu'à trouver ceux qui donnent le résultat attendu.

S'il y a N choix pour la première clé et autant pour la seconde (il y a un peu moins de choix en réalité pour k_2 grâce à la condition de primalité), une telle recherche exhaustive conduira à examiner N^2 combinaisons, ce qui ne sera pas raisonnable pour $N = 2^{25}$ comme c'est le cas ici.

---

3. `https://codingup.socinfo.fr/1024/24/pb/chiffrement.c`.
4. `https://codingup.socinfo.fr/1024/24/pb/message1.txt`.
5. Cf. https://fr.wikipedia.org/wiki/Inverse_modulaire[`https://fr.wikipedia.org/wiki/Inverse_modulaire`

Ici, la succession des deux méthodes de chiffrement (on parle de *surchiffrement*) constitue la faille du système.

Appelons F_1 la première méthode de chiffrement (par ou exclusif), F_2 la seconde (par multiplication modulo 2^{25}), m le message en clair, b le message chiffré à la première étape et c le message chiffré final :

- $b = F_1(m, k_1)$;
- $m = F_1^{-1}(b, k_1) = F_1(b, k_1)$ (propriété du ou exclusif) ;
- $c = F_2(b, k_2)$;
- $b = F_2^{-1}(c, k_2) = F_2(c, k_3)$ avec k_3, l'inverse de k_2 modulo 2^{25} (déchiffrer ce qui est chiffré avec k_2 revient à chiffrer avec k_3, l'inverse modulaire de k_2) ;
- si c est le chiffré de m avec les clés (k_1, k_2), alors $F_1(m, k_1) = F_2(c, k_3) = b$.

Ces éléments sont résumés sur la figure 11.

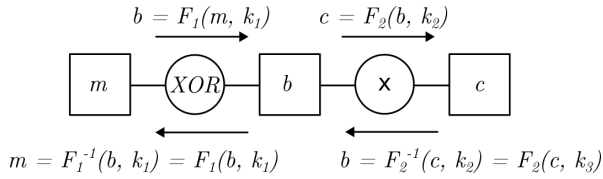

Fig. 12. Les deux étapes du chiffrement.

Nous allons présenter deux méthodes pour résoudre le défi. La première est plus générale (dès qu'on a un texte connu, clair et chiffré, et un surchiffrement), et la seconde pourra s'appliquer car on dispose en plus de deux blocs indépendants dans le texte connu.

Nous allons présenter deux méthodes pour résoudre le défi. La première est plus générale (elle s'applique parce que le texte en clair est connu et qu'il y a du surchiffrement), et la seconde pourra s'appliquer car on dispose en plus ici de deux blocs indépendants dans le texte connu.

Première méthode

L'idée générale est de trouver des couples (k_1, k_3) tels que $F_1(m, k_1) = F_2(c, k_3)$. Stockons dans un ensemble E toutes les valeurs $F_1(m, k_1)$ calculées en faisant varier k_1. Puis recherchons des valeurs possibles de k_3 en calculant $F_2(c, k_3)$ et en vérifiant si cette valeur est dans E (il faut pouvoir tester l'appartenance en temps moyen constant, ce qui sera le cas si E est représenté par une structure de données adéquate[6]). Dès lors, on aura trouvé une paire de clés qui déchiffre c en m. Toutes les paires de clés (k_1, k_3) convenables seront obtenues en un temps espéré $\Theta(N)$ au lieu de $\Theta(N^2)$.

6. Les structures de données Python dictionnaire ou ensemble font l'affaire.

Ce calcul révèle (en 3 ou 4 minutes sur une machine de bureau avec un programme Python), qu'il y a 1280 paires de clés qui transforment `MA.AYTBUAY` en `EXTERMINER`.

En voici par exemple deux :

– `CTM?J` et 7 471 149

– `PMTAZ` et 27 394 003

On peut à présent utiliser ces 1280 paires de clés sur le texte à décrypter, et vérifier quel résultat est intelligible. Il est assez fastidieux de contrôler ça manuellement, mais on peut par exemple trier les 1280 résultats obtenus par nombre de `E` (ou nombre d'espaces) décroissants, le bon résultat sera très probablement dans les premiers. Cette dernière partie prend un temps négligeable comparé au reste et le texte déchiffré est le premier de la liste (que l'on trie en utilisant `E` ou espace).

Cette première méthode est assez générale, et fait partie d'une des énigmes (avec des méthodes de chiffrement différentes) présentées dans ce livre[7].

Il est envisageable de n'utiliser que les 5 premiers caractères des messages clairs et chiffrés (un seul bloc), ce qui réduit le temps de recherche des clés potentielles à un peu plus de deux minutes, mais nous laisse avec plus de 16 millions de paires de clés et donc autant de messages à trier pour afficher en premier ceux qui correspondent *a priori* à un texte déchiffré.

Seconde méthode

On utilise ici le fait que le texte clair et le texte chiffré correspondant sont connus pour deux blocs indépendants m_1 et m_2 (`EXTER` et `MINER`), et que, par ailleurs, si $F_1(m, k_1) = b$, alors $F_1(m, b) = k_1$ (propriété du ou exclusif). Il est alors possible de réaliser le calcul suivant pour chaque valeur à tester de k_3 :

– calculer $b_1 = F_2(c_1, k_3)$ et $b_2 = F_2(c_2, k_3)$;

– calculer $k_1 = F_1(m_1, b_1)$ (valeur de k_1 telle que $b_1 = F_1(m_1, k_1)$) ;

– si k_1 est égal à $F_1(m_2, b_2)$, on a trouvé une paire de clés (k_1, k_3).

Par cette méthode, on retrouve les 1280 couples des clés en un peu moins de deux minutes.

c0d1ngUP team :
Emmanuel Laizé, université de Poitiers (UP),
Freddy Légé (UP), Laurent Signac (UP),
Benoit Tremblais (UP), Loïc Restoux, Caroline Tartary.

7. Pascal Lafourcade et Malika More, *25 énigmes ludiques pour s'initier à la cryptographie*, Dunod, 2021, 1re édition.