Solution à : Saturn Cipher.

La console demande l’entrée d’un bloc en hexadécimal de seize octets (32 caractères).

Première option :

Un message libre d’entrée est chiffré en clair connu de 16 octets (clé 128 bits) ; le résultat est restitué.

L’opération peut être répétée autant de fois qu’on veut.

Deuxième option :

le programme renvoie la version encryptée du flag est retourné : le calcul est basé sur une clé aléatoire d’une longueur égale à seize octets qui est donc indéchiffrable en « temps réel ».

L’encryptage réalise des opérations complémentaires de rotation et de permutation.

Les opérations dépendent a priori de la clé. Une faille de l’algo est repérée avec le clair connu. Si un caractère seul change en entrée, la chaîne encryptée est quasi identique. Cela signifie qu’il n’y a pas de diffusion. En plus, dans une même session, si un même caractère change en entrée, on obtient : une chaîne cryptée qui se modifie toujours au même endroit. Le challenge est sensible à une attaque octet par octet (caractère par caractère).

Par exemple, on part d’un clair connu à « 00000000000000000000000000000000 ».



On change un « 0 » en position n. On voit ce qui change dans la chaîne chiffrée. Quand on modifie le n ième caractère de la chaîne de départ, cela nous modifie le p ième caractère de la chaîne d’arrivée.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Ce n’est pas l’algo d’AES ici :

**Transformation et diffusion**

Supposons que seul l'octet en position **b₀** (coin supérieur gauche) est non nul dans l'état initial. Après plusieurs tours de transformations AES, cet octet influencera d'autres positions dans l'état. Par exemple, après deux tours, l'influence de **b₀** peut se propager à la position **b₁₀**.

Voici une représentation simplifiée de cette diffusion :

État initial : État après transformations :

[ b₀ b₄ b₈ b₁₂ ] [ \* \* \* \* ]

[ b₁ b₅ b₉ b₁₃ ] [ \* \* \* \* ]

[ b₂ b₆ b₁₀ b₁₄ ] [ \* \* b₁₀ \* ]

[ b₃ b₇ b₁₁ b₁₅ ] [ \* \* \* \* ]

Dans cet exemple, l'octet **b₀** a une influence sur l'octet **b₁₀** après les transformations. Cela illustre la propriété de diffusion, où la modification d'un seul octet d'entrée affectera plusieurs octets de sortie.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

On change maintenant le n ième caractère de la chaîne de départ par d’autres valeurs et on boucle : jusqu’à la valeur à la p ième position correspondante à la chaîne du flag (alignement sur n positions).

On a ainsi retrouvé la valeur du flag en position n. Il n’y a plus qu’à changer la valeur à la position n jusqu’à ce que la valeur donne le flag.

Faire tourner les valeurs du départ : 00000000000000000000000000000001 pour obtenir un seul caractère changé à l’arrivée en chaîne.

On peut donc tester donc, un à un, par caractère, si c’est le bon : ce n’est pas une diffusion où, si l’on changeait un caractère, toute la chaîne ou grande partie de celle-ci, serait changée en bout de course.

A chaque redémarrage de serveur (lancement d’une session : netcat), la clé de session est changée.

La dernière étape est de rejouer ce script sur le serveur netcat pour obtenir ainsi le vrai flag en HEXA.





