Devoir 1

Notes

Le code présenté est retrouvable sur ssh: mot de passe FDS8EbKiDNoJh2QN k

I) Chiffre de César

On lance le code si dessus 3 fois: (`make run part=1`) et on observe les sortie. On obtient quelque chose comme:

```
              -------PART 1-------
                     key: 5
    msg before: coding w/ freebsd style :)
      cypher: htinsl b/ kwjjgxi xydqj :)
     msg after: coding w/ freebsd style :)
                     key: 5
  msg before: ceciestlemessageclairadechiffre
    cypher: hjhnjxyqjrjxxfljhqfnwfijhmnkkwj
   msg after: ceciestlemessageclairadechiffre

              -------PART 1-------
                     key: 15
  msg before: ceciestlemessageclairadechiffre
    cypher: rtrxthiatbthhpvtrapxgpstrwxuugt
   msg after: ceciestlemessageclairadechiffre

              -------PART 1-------
                     key: 3
  msg before: ceciestlemessageclairadechiffre
    cypher: fhflhvwohphvvdjhfodludghfkliiuh
   msg after: ceciestlemessageclairadechiffre
```

J'ai laissé le premier message pour monter que l'on peut quand m^eme envoyer des symboles. Je l'ai enlevé après par soucis de place / redondance.

De leurs c^oté, Alice et Bob voient la clef key, le message msg et le cryptogramme cypher.

I) a)

Dans cette situation, Eve recevrait uniquement le cypher. On voit que quand on a des symboles autre que des lettre, il devient facile de voir des motifs, des mots. De plus, une analyse fréquentielle nous montrerais qu'il peut s'agir de texte français. Comme Eve n'a pas la clef, elle pourrait alors éssayer de bruteforce toutes les clefs du chiffrement de césar, ce que lui retournerait une solution rapidement (il n'y a que 26 clef possible).

II) OTP

II) a)

Comme on veut coder un message en char (1 octet) sur 512 bits (64 octets), j'ai pris la liberté de tester avec un message de cette taille, plutot que 32 octets.

Pour faire l'OTP, on fait un `XOR` entre le message et une chaine de 512 bits aléatoire. En faisant ceci, on doit faire attention de bien terminer la string avec un nullbyte, pour que notre programme fonctionne bien. Cela nous laisse donc 63 octets pour faire un message.

J'ai représenté dans mon programme les valeurs des cryptogrammes et messages en clair ainsi que leur représentation en hexa, pour pouvoir comparer et montrer que nous avons bien fait un `XOR` Cependant, le compilateur à décider l'allouer des int, plutot que des char, et nous nous retrouvons avec des choses de la forme de FFFFFFD6

```
-------PART 2-------
key: @2@@@@@@v@@@@_\@jXS@@9@#@@0@O|@1@T@2@@d@@@BM@Ɔ@wi@u@z@)
msg before: ceciestlemessageclairadechiffreilcomporte64charcestplutotlongxd
msg as hex: 63 65 63 69 65 73 74 6C 65 6D 65 73 73 61 67 65 63 6C 61 69 72 61 64 65
63 6
 6C 6F 6E 67 78 64
cypher: @W@@@@@@M>;@    42@@X@F@@Y@)@X@7h@B@@₫@P0@ⷮ.}@˝@@l@
cypher as hex: FFFFFF81 57 FFFFFFD6 FFFFFFBF FFFFFFBE FFFFFFAD FFFFFF99 FFFFFF91 13
FFFF
FCB FFFFFFC1 59 FFFFFF98 29 0E FFFFFFFA 58 FFFFFFC2 37 68 FFFFFFE7 42 FFFFFF9A
FFFFFFF5
FFFFFA3 FFFFFFCA FFFFFFBA 18 1D FFFFFF92 1A FFFFFFD6 1D 6C FFFFFF8D
msg after: ceciestlemessageclairadechiffreilcomporte64charcestplutotlongxd


-------PART 2-------
ˇ@@@@@@py@F)SG@@HP@-@D@@@       @@ƱZ@f@@@@ @c@@@
msg before: ceciestlemessageclairadechiffreilcomporte64charcestplutotlongxd
msg as hex: 63 65 63 69 65 73 74 6C 65 6D 65 73 73 61 67 65 63 6C 61 69 72 61 64 65
63 68 69 66 66 72 65 69 6C 63 6F 6D 70 6F 72 74 65 36 34 63 68 61 72 63 65 73 74 70
6C 75 74 6F 74 6C 6F 6E 67 78 64
cypher: @@9@P]?"@@;#@J@@@%@@@@j@@@@(@@@ۓ@@Ro@W@@l}~@ƀ@@z@@@
cypher as hex: FFFFFFD6 19 FFFFFFAE 39 FFFFFF93 50 5D 3F 22 FFFFFFC8 FFFFFFE0 3B 23
FFFFFFB8 4A FFFFFF96 FFFFFFF3 FFFFFFE4 25 FFFFFFD5 FFFFFFD0 FFFFFFD8 7F FFFFFFAC 6A
FFFFFF90 FFFFFFFB FFFFFFA0 FFFFFFD7 28 FFFFFF86 0F FFFFFFBA FFFFFFD3 FFFFFFD8
FFFFFFA0 FFFFFFA4 FFFFFF8F 52 6F FFFFFFE0 FFFFFF93 57 FFFFFFB6 17 FFFFFFF1 FFFFFFBB
6C 7D 7E FFFFFFBF FFFFFFCA FFFFFFAB FFFFFF93 FFFFFFF0 FFFFFFBF FFFFFFAA 7A FFFFFFF9
FFFFFFE1 17 01 FFFFFF92
msg after: ceciestlemessageclairadechiffreilcomporte64charcestplutotlongxd


-------PART 2-------
key: @@@
        ₫-@.)@<@W9~vs`@
                        uN@@5@#
                               @@@@@@m@@Ek@@@J8@@@ 6k@(w@
msg before: ceciestlemessageclairadechiffreilcomporte64charcestplutotlongxd
msg as hex: 63 65 63 69 65 73 74 6C 65 6D 65 73 73 61 67 65 63 6C 61 69 72 61 64 65
63 68 69 66 66 72 65 69 6C 63 6F 6D 70 6F 72 74 65 36 34 63 68 61 72 63 65 73 74 70
6C 75 74 6F 74 6C 6F 6E 67 78 64
cypher: @fûnoe@@@@]Z@[@4U@n&@S@Fb@@@@@@@y@@&@@@t9L@ə@OBuj@P
```

```
cypher as hex: FFFFFFB5 66 FFFFFFC3 FFFFFFBB 6E 6F 65 FFFFFFB6 FFFFFFE8 40 FFFFFF84
5D 5A FFFFFF80 5B FFFFFFE2 34 55 1F 1F 01 01 FFFFFFCB 6E 16 26 FFFFFFE3 FFFFFF83 53
FFFFFFB2 46 62 FFFFFFAF FFFFFFA0 FFFFFFB1 FFFFFFA2 FFFFFF90 FFFFFF80 FFFFFFDB 19 79
FFFFFFBC FFFFFFAF 26 03 FFFFFFB6 FFFFFFBE FFFFFFA1 74 39 4C FFFFFFF4 FFFFFFC7
FFFFFF9D FFFFFFFB 4F 42 75 6A 05 FFFFFFBE 50 13
msg after: ceciestlemessageclairadechiffreilcomporte64charcestplutotlongxd
```