## Solution à : R1R2.

Le script fourni par l'énoncé (encrypt.py) montre le mot de passe chiffré / encrypté à la fin.

- 1. **Séparation du mot de passe** : (rechargement de l'empreinte HEX\_CIPHER (HEXA)). password[::2]  $\rightarrow$  b (caractères d'indice pair), password[1::2]  $\rightarrow$  c (indices impairs).
- 2. **Transformation en entiers**:  $b_int$ ,  $c_int = int.from_bytes(..., 'big') puis on impose <math>b_int > c_int$ .
- 3. Constantes utiles:
  - d = b\_int + c\_int
     e = b\_int \* c\_int
- 4. Trois couples (x, y) sont publiés :

$$y = x^2 - d \cdot x + e$$
.

Chaque couple est sérialisé dans 1023 bits via la fonction-ci (1 bit de signe, 511 bits pour |y|, 511 bits pour x) puis *padded* sur 128 octets.

Connaître trois couples (x, y) suffit donc à retrouver d et e (verrou de degré 2), puis b\_int et c\_int (via les racines d'un polynôme), et enfin le mot de passe original par un entremêlement des deux moitiés. Le flag est concaténation des 2.

```
print("=== Moitiés ASCII ===")
          print(f" b_str = {b_str!r}")
         print(f" c str = {c str!r}\n")
          flag = "".join(a + b for a, b in zip(b str, c str)) \
                     + b_str[len(c_str):] + c_str[len(b_str):]
PROBLEMS 8 OUTPUT DEBUG CONSOLE TERMINAL PORTS
=== Constantes obtenues ===
  d = b + c = 23698000211972116834885198358973470299714808136601656247983712977870923160734
                    = 2013527388962450148751409699527657830636009572701640801374083888144490941616272806053222385729
081550741560410237956433004125791049982477311842523893789
=== Racines du polynôme ===
   \Delta \ = \ 55354110449078069384339153808601689049870162164144747553386622106364816891247102642577617038307383695
6828949230976890713088368606522362532891530703843600
   b int = 23612727241664409124333069945237177781262596061818334584244137178318328851837
   c int = 85272970307707710552128413736292518452212074783321663739575799552594308897
=== Moitiés ASCII ===
   b_str = '44T{1_0PBMa7d5rto_Rga_CI83!2-0m}'
   c str = '0CFBpbu 6!U03cU1Np06Ma7vtd31Bo!
=== Flag final ===
   404CTF{B1p b0uP B6M!aU70d35crUt1oN pR0g6aM aC7Iv8t3d!321-B0om!}
 19 V HEX CIPHER =
          40f1b6e577b2bb6aa703387a15d2738ad50c795972342bdbb4b32946bcf7b72f
          0c4df9ca82064c5b97e3e5013732439d6139195456b94e581b7a22f1510f926c
          fc3820a17947e71689812f6113dd3893250a14320a8f49c46bde754a188efd30
          940d748f2696de5597ec5df6d12826fc2b37d8e926af7a39afe74cb0950460da
          la33112d89029e1a9334ea4c19d36cab027d4b360f240139de4ebd58ebfb0568
PROBLEMS 8 OUTPUT DEBUG CONSOLE
                                                         TERMINAL PORTS
=== Recherche des combinaisons ===
 \begin{tabular}{ll} [\c Algorithm]{ll} & Algorithm 
  e=201352738896245014875140969952765783063600957270164080137408388814449094161627280605322238572908155074156
0410237956433004125791049982477311842523893789
       828949230976890713088368606522362532891530703843600 b=23612727241664409124333069945237177781262596061818334
584244137178318328851837 c=85272970307707710552128413736292518452212074783321663739575799552594308897
       b str='44T{1 OPBMa7d5rto Rga CI83!2-0m}' c str='0CFBpbu 6!U03cU1Np06Ma7vtd31Bo!
       FLAG Candidat : 404CTF{Blp_b0uP_B6M!aU70d35crUt1oN_pR0g6aM_aC7Iv8t3d!321-B0om!}
FLAG = 404CTF{B1p b0uP B6M!aU70d35crUt1oN pR0g6aM aC7Iv8t3d!321-B0om!}
```