Assignment 4

Due date: see Canvas

Please complete this assignment (100 pts total + bonus) and submit your report/program code on Canvas (all files compressed in one `.zip` without the `.bin/.txt` files)

1. A program needs to be developed such that fault attacks on a RSA-CRT implementation can be exploited to reveal the original RSA primes of a 1024 bit RSA key. The fault was previously generated during one CRT exponentiation and caused a faulty s' of the RSA-CRT while computing the digital signature. The wrong result s' is the important input parameter for the program. Other input parameters for the program are the public RSA parameters, moduluns N and exponent e, in addition to the correct result s and the message m to be signed. Two different use cases should be considered. (80 pts)
   a) the RSA modulus N and both the correct result s and a faulty result s' of an RSA-CRT are used (Bellcore) (40 pts)

```
q=0x9f44ddf28f05904455669a629df988adf203812f56aa8047c7db9bb7b4e61dd67b027e80
d8700a77471943cc76370ced07056ef808a12b2a467c159e586c33
```

```
p=0xf9555b790d60dcb3fdcdf464b88ab7bb629bfce037f4154927df19fcdb1b4c7327d41b17
d848455cffbda7e8080c08600be3af126df6c481ab25da70bec471c0fb
```

   b) the RSA modulus N, the public exponent e, the faulty result s', and the message m are used (Lenstra) (40 pts)

```
p=0xd4693216ca3210f1491477d556e709141f6b5ea57e8b64a51011190d607b6b92a60185
7e4ad26e2b45123804ebdd08ccd15b0e50edcdc8754d5b2bb99dc8286087
```

```
q=0xeacd987fce4c2815b8e1f6557a4120cd822763baa732e6fbd2d35d61b85f8278263ce068
cddf6099ba885cda0b4ed1c2374de5d34b265fec3358611905ae81
```

```
d=0x6f7345e591342f162230a8b392814b0f4f80268e7008b129cf0c4c009ad4cce91e6a3f1d
2a5eb72ed86e55b079a8a2963248640819b1121f0411d0ba1b1647445bf2438288738d9eca
ef90ed7b3a4d1170f22f28cc60396854b9508df0cc39397bbc4eae35428edb25416b6370bda
32bc8d58ac6fceef0b3d94a0d65c7ef1501
```

   All needed input parameters are provided in two separate ASCII files. The hexadecimal representation starts with the most significant bit and ends with the least significant bit. Determine the RSA primes from the input file `fault1.txt` and `fault2.txt`.

   Your program should output the original RSA primes. Note: you do not have to parse the files but can copy-paste the respective values into your script.

2. Conduct Differential Fault Analysis (DFA) on AES-128. You are given a set of three files `ptext.bin`, `ctext.bin`, and `ftext.bin`, corresponding to the plaintext, ciphertext, and (potentially) faulty ciphertext respectively. Note that faults are always injected before the "MixColumns" operation in the 9th round of AES, and each fault only impacts at most one byte. (70 pts)
   a) Having a reliable fault injection setup is critical to successfully executing DFA. From

the glitch data provided, how many glitches are successful? Does that seem like a high success rate? (5 pts)

The number of successful glitches is 402

Success rate: 40.20%

40.20% success rate is medium，I think the 40.20% success rate shows that the fault injection setup is relatively reliable

b) To perform the DFA attack described by Piret and Quisquater, we will need to find multiple ciphertext/faultytext pairs. How many total pairs will we need? By parsing the provided files, find enough pairs to complete the attack and output them here in hex format. Can anything happen that might cause you to need more pairs? (15 pts)

Based on the results, I think theoretically we need at least 8 ciphertext/fault ciphertext pairs, because each pair can provide some information about the key.

If we assume that the fault only occurs in the first row of the AES state matrix, rather than at any position globally, the range of fault injection will be limited. This may require more fault pairs to ensure that the key bytes of each column can be correctly extracted.

```
Column column_1:
  Pair 1: Ciphertext: 23db99defc61747b8c5f36651b7f261e, Faulty Ciphertext: 33db99defc6174f68c5fa3651b2e261e
  Pair 2: Ciphertext: e5992aabc2d122a6de020bcd537ead0f, Faulty Ciphertext: 95992aabc2d1229ade0222cd5356ad0f
Column column_2:
  Pair 1: Ciphertext: 729002061e1383e8fca65142e902cab4, Faulty Ciphertext: 72d70206441383e8fca651e9e90243b4
  Pair 2: Ciphertext: 5cc1b561e70bd538b1761bed86b4dc26, Faulty Ciphertext: 5c89b561f00bd538b1761bf186b49126
Column column_3:
  Pair 1: Ciphertext: bb4f38f6cae5b63bb130e8db5db06342, Faulty Ciphertext: bb4fcaf6caddb63b7530e8db5db06355
  Pair 2: Ciphertext: ecb49e914cac9e83f1e1f95be0874d10, Faulty Ciphertext: ecb4d4914c4c9e83ede1f95be0874d48
Column column_4:
  Pair 1: Ciphertext: b6dc67dbc6f7632c1a160ea4c74485c7, Faulty Ciphertext: b6dc6715c6f7602c1a240ea4aa4485c7
  Pair 2: Ciphertext: 3bcdd2b4dc96676afbd8718d4ea9bf0d, Faulty Ciphertext: 3bcdd219dc96196afbae718d71a9bf0d
```

c) Simple Piret and Quisquater DFA: We will provide pairs with glitch in the first byte. Recover 4 bytes of the key (20 pts)

```
The key bytes for the simple attack are: [(168, 138, 164, 45)]
```

d) Full Piret and Quisquater DFA: using the provided encryption/glitch data, recover the entire round 10 key (20 pts)

[168, 73, 55, 172, 53, 213, 50, 45, 93, 35, 164, 0, 170, 138, 46, 198]

```
First set of key bytes for Phase 1: [(168, 138, 164, 45)]
Second set of key bytes for Phase 2: [(53, 73, 46, 0)]
Third set of key bytes for Phase 3: [(93, 213, 55, 198)]
Fourth set of key bytes for Phase 4: [(170, 35, 50, 172)]
```

e) Recover the original (first round) key and use it to decrypt the following (hex encoded) secret message: `2a92fc6ad8006b658f49062c2843ad99` (10 pts)

```
The original key is: b'=\x83\xa4\x01t\xa3Xg;l=\x99\xdcS\x92\xc3'
The secret message is: DFAIsAFunAttack!
```

Please write all your programs in one of the following languages/environments: Python/Jupyter, Rust, C/C++, Matlab/Octave, Java. Your `.zip` file should contain your code, instructions how to make it run (if needed), the figures, a report, etc.