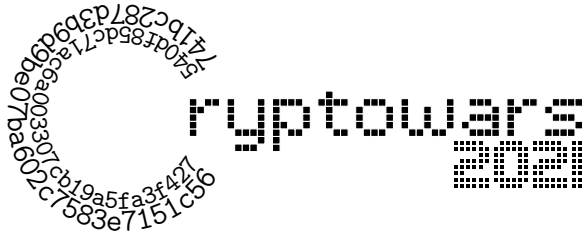


October 25, 2021, 14:00 (CEST) – October 31, 2021, 23:59 (CET)

Rules

- Requests are marked with ▶.
- Every challenge is identified by the **sub**section it belongs to. The solution file must be named with the subsection number concatenated to `.txt` extension. For instance, `2.1.txt`.
- Every challenge will be evaluated 1 point.
- Ties will be resolved by the submission time (the former, the better).

Contents

1 Asymmetric

1.1 More of the same

The file named `1.1.txt` contains three RSA public keys, i.e. three pairs of numbers (N, e), N being the modulus and e the public exponent.

- For each public key, find p and q such that $N = p \cdot q$. Find also d such that $ed = 1 \pmod{\varphi(N)}$, φ being the Euler's totient function.

1.2 Marriage paranoia

The wonderful day is coming: Alice is spreading her wedding participation! To the four closest friends, she attaches some verses from a famous literary work. Exceeding with her paranoia, Alice encrypts her message M_b through

$$C_i = (M_b + 3^i)^e \pmod{N_i}$$

using her friends' public parameters. The `1.2.bride.txt` file contains the public exponent e, N_i and C_i for $i = 1, 2, 3, 4$.

- Recover M_b.

Also Bob, the groom, wants to look like an intellectual person, so, he sends to his friends a poetry. He encrypts the poetry M_g through

$$C_i = M_g^e \pmod{N_i}.$$

using his friends' public parameters. The `1.2.groom.txt` file contains the public exponent e, N_i and C_i for $i = 1, 2$.

- Recover M_g.

1.3 Repetita iuvant

The file named `1.3.txt` contains instances of *ECDLP*.

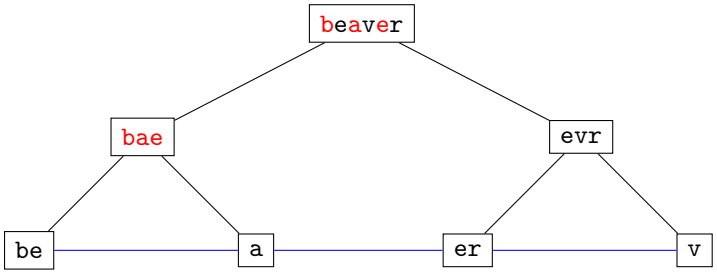
- For each instance, recover the k such that $Q = kP$ on the curve $y^2 = x^3 + ax + b$, where a and b are defined mod p.

1.4 Computer sloth

Bernardo is working as a system security expert in a big video-game company. Shortly before the end of the work shift, the general manager has established that every video-game will be digitally signed using ECDSA. Bernardo has to write as fast as possible the code that generates the elliptic curves. He cannot wait to go home, so, he searches for some parameters and he decides to use those in file `1.4.txt`.

Bernardo asks Evelina to check the code and she gets Q_i using P_i $(i = 1, 2)$ (check again file `1.4.txt`). The very next day, Evelina forces Bernardo to generate a new code, because she has easily recovered the secret key k.

Figure 1: Mechanism of the *Beaver* cipher.



▶ Which is k?

2 Classic

2.1 Zero

A mysterious character is currently in an unknown city. He says the content of the file `2.1.txt`.

▶ Who is talking?

2.2 Love

We start by recalling a cipher: *Beaver*. Encrypting a message with the *Beaver* cipher consists in creating a binary tree by splitting the letters of the message between "letters in odd position" (labeling the child on the left) and "letters in even position" (labeling the child on the right). The process iterates until the nodes contain only one or two letters. Reading the leaves from left to right, one gets the cipher-text.

As an example, the encryption of the word `beaver` is shown in figure 1. The resulting cipher-text is `beaerv`.

The message contained in the file `2.2.txt` represents a dialogue involving two characters. It has been encrypted in this way:

- the key is formed by a concatenation of a word w with the remaining letters in the alphabet kept sorted;
- the i-th letter of the alphabet is substituted by the i-th letter of the key.

You know that w is `tnmio`, encrypted using *Beaver*.

▶ What is w?

▶ Who are the two characters involved in the ecrypted message?

2.3 Crowns

Alice sends an encrypted message to Bob using her version of the *Playfair* cipher. In particular, Alice has forgotten the letter `W`. Eve recovers four chars of the secret key `xxxIDER`, but she still need the first three (denoted by `x`).

The encrypted message received by Bob is `PCMHQBJMQTXB`.

- ▶ Which is the plain-text sent by Alice?

2.4 Students

Alice is starting a first course in Cryptography. She wants to send an encrypted message to Bob using the Hill cipher. In particular, Alice uses the alphabet $A = 0, \ldots, Z = 25$ and a random secret key, which is `MYBEZHWFR`. Unfortunately, it is not a good choice... You know that the plain-text sent by Alice is `PTO`.

- ▶ Find another 3 char string whose encrypted message is the same.

3 Hash

3.1 Italians do it better

SHA-1 is a hash function that evolves through 80 rounds and a final digest consisting of 160 bits. Consider the states from the 17th round (notice that rounds run from 0 to 79) to the final digest.

- ▶ Find which couples of different italian girls (see `3.1.txt` file) has states (round outputs or final digest) with the maximum collision (i.e. maximum Hamming weight collision).

The content of the solution file must be in the following format:

`w0|r0|s0|w1|r1|s1`

where

- `w0`, `w1` are the girls, i.e. a string, remembering that we want `w0` different from `w1`;
- `r0`, `r1` are the rounds, i.e. an integer greater or equal to 16, number of rounds starts from 0;
- `s0`, `s1` are the states, a string representing the hexadecimal value without the leading `0x`.

3.2 Mexican culinary art

See the file `3.2.txt`.

- ▶ For each string s_i contained in the file, find n_i strings x whose length is l_i, such that the leftmost nibbles of $f(s_i \parallel x)$ are h_i such that
  - f the SHA-1 hash function;
  - $\parallel$ the concatenation of strings.

The content of the solution file must have the following format: the first l_1 lines are the sample for the string s_1, next l_2 lines are samples for the string s_2 and so on. Empty lines can be used if you do not reach the target l_i.

4 Stream ciphers

A Linear Feedback Shift Register (LFSR) of length ℓ with feedback polynomial

$$x^\ell + c_{\ell-1}x^{\ell-1} + c_{\ell-2}x^{\ell-2} + \cdots + c_1 x + c_0 \in \mathbb{F}_2[x]$$

is a register with ℓ bits $[s_{\ell-1}, s_{\ell-2}, \ldots, s_1, s_0]$, which, at each clock, outputs the rightmost bit and updates its inner state in the following way

$$[s_{\ell-1}, s_{\ell-2}, \ldots, s_1, s_0] \mapsto [s_\ell, s_{\ell-1}, \ldots, s_2, s_1]$$

where

$$s_\ell = c_0 s_0 \oplus c_1 s_1 \oplus \cdots \oplus c_{\ell-2} s_{\ell-2} \oplus c_{\ell-1} s_{\ell-1}.$$

Remark. The two following sections are independent.

4.1 Fantastic!

We are going to explain how a *nonlinear combiner generator* works.

Consider a stream cipher with four LFSRs with feedback polynomials:

$$\begin{aligned} &x^{23} + x^5 + 1, \\ &x^{17} + x^3 + 1, \\ &x^{13} + x^4 + x^3 + x + 1, \\ &x^{11} + x^2 + 1. \end{aligned}$$

We consider the inner state of the cipher as the concatenation of the inner states of the four registers $\mathrm{LFSR}_1, \mathrm{LFSR}_2, \mathrm{LFSR}_3, \mathrm{LFSR}_4$.

At each clock $t \geq 0$, each LFSR outputs the rightmost bit. Denote by x_i the output of LFSR_i, for $i = 1, \ldots, 4$. These four bits form the input of the following nonlinear Boolean function:

$$g(x_1, x_2, x_3, x_4) = x_1x_2 + x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4$$

to produce the output z_t. Then the four LFSRs update.

The keystream consists of the collection of bits returned by g, namely $\{z_0, z_1, z_2, \ldots\}$.

In figure 2, the mode of operation of the above described stream cipher is displayed.

To verify the correctness of your implementation, produce a keystream of length 128 bits from the initial state `12345678` (consider it as a string of ASCII characters). Then compare the result with the content of the file `4.1.ex.txt`.

- Each line in the file `4.1.txt` contains a keystream obtained via the nonlinear combiner generator. For each keystream, compute the initial state, decompose it in bytes and convert it in ASCII. The resulting strings are the one you have to send.

Figure 2: A stream cipher with a nonlinear combiner generator.

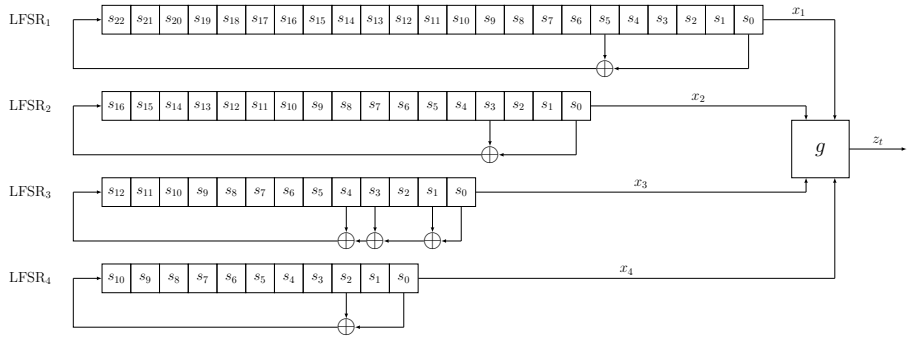
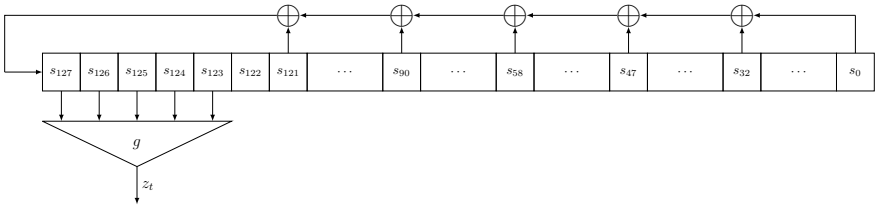


Figure 3: A stream cipher with a nonlinear filter generator.



4.2 Think abstract

We are going to explain how a *nonlinear filter generator* works.

We have one LFSR with feedback polynomial:

$$x^{128} + x^{121} + x^{90} + x^{58} + x^{47} + x^{32} + 1.$$

At each clock $t \geq 0$, we give the inner state $[s_{127+t}, s_{126+t}, \ldots, s_{1+t}, s_t]$ as the input of the Boolean function

$$\begin{aligned} g = x_{127}x_{126}x_{125} + x_{127}x_{126} + x_{127}x_{125}x_{124} + x_{127}x_{125}x_{123} \\ + x_{127}x_{124} + x_{127} + x_{126}x_{125} + x_{124} + x_{123} \end{aligned}$$

to produce the output z_t. Notice that the Boolean function only uses the 5 leftmost bits of the inner state. Then the LFSR updates. The keystream consists of the collection of bits returned by g, namely $\{z_0, z_1, z_2, \ldots\}$.

In figure 3, the mode of operation of the above described stream cipher is displayed.

To verify the correctness of your implementation, produce a keystream of length 256 bits from the initial state `123456789ABCDEFG` (consider it as a string of ASCII characters). Then compare the result with the content of the file `4.2.ex.txt`.

- The file `4.2.txt` contains a keystream obtained via nonlinear filter generator. Recover the initial state, decompose it in bytes and convert it in ASCII. The resulting string is the one you have to send.

Figure 4: Pseudo-code for the cipher.

```
Cipher(byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
begin
    byte state[4,Nb]

    state in

    AddRoundKey(state, w[0, Nb-1])

    for round = 1 step 1 to Nr-1
        SubBytes(state)
        ShiftRows(state)
        MixColumns(state)
        AddRoundKey(state, w[round*Nb, (round+1)*Nb-1])
    end for

    SubBytes(state)
    ShiftRows(state)
    AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1])

    out = state
end
```

5 Symmetric

5.1 X-files

We are now considering an AES-128 scheme that runs just for 4 rounds. Specifications can be found in the NIST site. Figure 4 describes the scheme implementation. In our case, it is `Nr=4`.

- Among the files of the challenge there is only one encrypted with 4 rounds AES-128. Find it.

5.2 Save the last key

See the `5.2.txt` file.

- Given plain-texts and corresponding cipher-texts, recover the key of the last round.

Remark. Called k_0 the first added key and k_i the key of the i-th round, we are searching for k_4.