Homework 4 - Elliptic Curves and Symmetric Key Cryptography

Cryptography and Security 2020

- You are free to use any programming language you want, although SAGE is recommended.
- Put all your answers **and only your answers** in the provided GROUP-answers.txt file. This means you need to provide us with all `Q` values specified in the questions below. You can download your **personal** files from the following link:
  `http://lasec.epfl.ch:80/courses/cs20/hw4/index.php`
- You will find an example parameter and answer file on the moodle. You can use this parameters' file to test your code and also ensure that the types of `Q` values you provided match what is expected. **Please do not put any comment or strange character or any new line** in the .txt file.
- We also ask you to submit your **source code**. This file can of course be of any readable format and we encourage you to comment your code. Notebook files are allowed, but we prefer if you export your code as a text file with a sage/python script.
- The plaintexts of most of the exercises contain some random words. Don't be offended by them and Google them at your own risk. Note that they might be really strange.
- It is sufficient if one of the students from each group submits the answer/code.
- We might announce some typos/corrections in this homework on Moodle in the "news" forum. Everybody is subscribed to it and does receive an email as well. If you decided to ignore Moodle emails we recommend that you check the forum regularly.
- **Do not submit the test-case solution!** This causes us a lot of trouble to grade!
- The homework is due on Moodle on **Friday the 27th of November** at 23h59.

Exercise 1 Stop the counter!

Old Joe and Papy McDonald are competing in the annual beauty contest of their retirement home. This year, their home secretary decided to authenticate each ballot with ECDSA to prevent fraud. More formally, here is how the election takes place:

1. Each pensioner goes to a nurse and announces her/his choice after identification.
2. Each ballot of the form *X voted for Y* is signed with ECDSA and put into the electronic ballot box.
3. Everyone can verify the authenticity of the ballots in the box by using the ECDSA public key.

The parameters used for the curve are those from the `secp256k1` standard. The curve is given by the equation $y^2 = x^3 + 7$ over the finite field $\mathsf{GF}(p)$. In your parameters file, you will find p, the subgroup generator G and the secretary's public key `pk`.

Question a)

After observing the first 30 ballots, Papy McDonald is convinced that a massive fraud is happening and asks you to verify all the ballots in order to prove it. You will find all the ballots in the variable `Q1a_arr` of your parameters file. You should verify all of them and report any fraudulent ballot under the `Q1a_msg` variable (**report only the message, not the signature**).

Question b)

Driven by this indisputable proof of unprecedented fraud, Papy McDonald decides to act swiftly. He manages to stop the electronic counter of the poll, which counts the number of ballots received. Then, as a final gesture to invalidate the contest, Papy asks you to forge a valid ballot for the message *I won this election, by a lot!* using the random value k he provided.

Fortunately for you, it turns out that the randomness used in the ECDSA signature is deterministically derived from a secret value and the counter. In addition, you will find in `Q1b_arr` two valid ballots that were published after the counter was stopped. Report the forged signature (r, s) under `Q1b_r` and `Q1b_s`. In addition, the k that must be used in the forgery is given in `Q1b_k` (note that k is given because we want a unique solution).

Hint: The following Sage code snippet shows how a message must be hashed and converted into an integer in our version of ECDSA.

```
>>> import hashlib
>>> m = 'covfefe'
>>> h_bytes = hashlib.sha256(m.encode()).digest()
>>> h = int.from_bytes(h_bytes, "big")
```

Exercise 2 Cryptomancer's Tale IV: In Pursuit of Perfection

The old evil Cryptomancer has tried many tricks to attain perfect secrecy and failed, but this time he is sure he has managed to achieve his lifelong desire.

After seeing how complicated the elliptic curve point addition is, he decides to generate long keys from small seed, using elliptic curves. As he also cares about distinguishability, and as Vernam cipher gives information about the length of the message, he uses padding to hide the length of the message.

His idea is as follows, let E/F_p be an elliptic curve with j-invariant 0 or 1728, and P be a fixed point on this curve. Let the function $\log_{-1}$ be such that $\log_{-1}(1) = 0$, $\log_{-1}(-1) = 1$, $\log_{-1}(0) = 0$, and let $k \in \mathbb{Z}$ be a random integer. The one-time secret key $K = K_0, \dots, K_\ell$ is computed with the following formula,

$$K_i = \log_{-1}\left(\tfrac{x((k+i)P)}{p}\right)$$

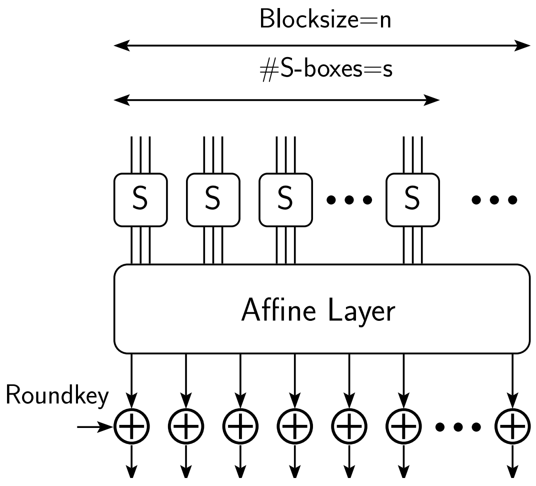
The accurate description of the encryption is given in the following algorithm.

```
**Input:** $p, M, P, \ell$
**Output:** $C, k$
$k \xleftarrow{\$} F_p$;
$r_1 \xleftarrow{\$} [1, \ell - |M|/8]$;
▷ padding with $r_1$, and $\ell - r_1 - |M|$ zero bytes
$M' \leftarrow 0x00^{r_1} || M || 0x00^{\ell - r_1 - |M|}$;
**for** $i \in \{0, \dots, \ell - 1\}$ **do**
    $K_i = 0$;
    **for** $j \in \{0, \dots, 7\}$ **do**
        ▷ Legendre symbol of the x-coordinate
        $K_i \leftarrow K_i + 2^{7-j} . \log_{-1}\left(\frac{x((8i+j+k)P)}{p}\right)$;
    **end**
    $C_i \leftarrow M'_i \oplus K_i$;
**end**
**return** $(C_0 || \dots || C_\ell, k)$;
```

In your parameters file you are given the characteristic of the field Q2_p, the cardinality of the elliptic curve E, Q3_n, and a ciphertext Q3_ct = (C,k). Knowing that the parameters ℓ and r_1 are badly chosen, you are asked to recover the plaintext Q3_pt which is a meaningful English sentence.

Exercise 3 MBCGA

After losing the beauty contest, our favorite protagonist Papy McDonald has decided to find some remedy by making Blockciphers networks great again. He decided to mimic what the AES S-box for his design. The block-size and key size in this block-cipher are equal to eachother and are both 128-bits. The round function of this block cipher looks as shown in the following diagram.



To simplify things he decided to keep s, the number of S-boxes in each round, to be 1. The k^{th} round Affine layer basically multiplies the state vector by the k^{th} Matrix L_k given in the cipher parameters, and the k^{th} layer roundkey is basically the master key multiplied by A^k, where A is the key-update matrix, also given in the parameters. The Sbox $S(x, y, z)$ is simply given by casting $0bxyz$ to $F_8 = \mathbb{Z}_2[x]/\langle x^3 + x + 1\rangle$ and inverting the element in the final field, a.k.a $F(a, b, c) = \text{coefficients}(ax^2 + bx + c)^{-1}$. You are asked to answer the following 3 questions about this cipher.

Question a)

In your parameters file you are given a plaintext Qa_pt, a master secret key Q3a_K, the affine matrices Q3a_R and the key update matrix Q3a_A. You are asked to implement McDonald's protocol and find the corresponding ciphertext Q3a_ct as a bit string, e.g. "1101101 . . . 01".

Question b)

In your parameters file you can find a plaintext Q3b_pt1, the corresponding ciphertext Q3b_ct1, encrypted using a 2-round variant of the McDonald cipher. You are also given the round affine matrices Q3a_R and the key update matrix Q3a_A. You are asked to decrypt the given ciphertext Q3b_pt1 and find the corresponding plaintext Q3b_pt2, which is a printable string.

Question c)

The task of this question is exactly the same with the previous one! But this time a 5-round variant of the cipher is used. Decrypt Q3c_ct2 and find the corresponding plaintext Q3c_pt2, which is a printable string.

Hint 1: Try to write the Sbox as a function from $\mathbb{Z}_2^3 \rightarrow \mathbb{Z}_2^3$.

Hint 2: What can we say about $S(a, b, c)$ if we know the value of $Maj(a, b, c)$, $Maj(*, *, *)$ being the majority function. Can you simplify the S-box given the majority?