Homework 2 - Prehistoric to Modern Crypto

Cryptography and Security 2019

- 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 SCIPER-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: `https://lasec.epfl.ch/courses/cs19/hw2/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. For instance, the variable `Q1_order` should be an integer, whereas `Q2` is an ASCII string. **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 textfile with a sage/python script.
- If you worked with some other people, please list all the names in your answer file. We remind you that you have to submit your **own source code** and **solution**.
- 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.
- The homework is due on Moodle on **Thursday the 24th of October** at 22h00.

Exercise 1 Order!

Let us start off with an easy exercise! Do not worry; there will be way harder exercises!

In your parameters file, you are given an almost 300-bits long prime p. IMPOSSIBLE! Consider the elements in $\mathbb{Z}_{p^2}^{\times}$. Some of these elements have odd orders, and some have even orders. Let r be the smallest, positive, non-trivial, odd order (non-trivial means $r \neq 1$).

I challenge YOU to find the smallest element in $\mathbb{Z}_{p^2}^{\times}$ which has order r (by smallest we mean if you consider the representative in $\{1, \ldots, p^2 - 1\}$).

In your solutions file, write this element in front of `Q1_element`, and the order in front of `Q1_order`. You can find the prime p in your parameters file.

Hint: Are all large numbers hard to factorize?

Exercise 2 The Adventures of the Crypto-Apprentice: Generalized Vernam Cipher With Key Expansion

A young apprentice cryptographer just enrolled in the course Cryptography and Security. Eager to get some hands-on experience, he refused to wait until the end of the course and wanted to apply some cryptography straight away. During the lecture, he was impressed by Vernam cipher which provides the perfect secrecy, but he found it is not practical since the key length should be equal to the length of the message.

During the weekend, the apprentice came up with a brilliant idea and he wrote his own version of generalized Vernam cipher, call it CA-Vernam. His idea was to generate a key stream by using the parity of multiples of a random element in a group. In CA-Vernam cipher, sender first selects an odd prime p and then an integer $a \in [1, p-1]$ to generate a key stream. Let $m = m_0 || \cdots || m_{2\ell-1}$ be a UTF-16BE encoded message to encrypt where m_i is a byte. Then, the encryption of m is $c = c_0 || \cdots || c_{2\ell-1}$ where $c_i = m_i \oplus k_i$ for $i \in \{0, \ldots, 2\ell - 1\}$ where $\oplus$ is byte xor operation and a byte k_i is defined as

$$k_i = k_{i,0} || \cdots || k_{i,7}$$

where a bit $k_{i,j} = ((8 * i + j + 1)a \bmod p) \bmod 2$, i.e. t-th bit of the key stream k is the least significant bit of $(t+1)a \bmod p$. The apprentice was so proud of his construction and asked you if you can break this CA-Vernam cipher. You received the ciphertext c_2 which is an encryption of unknown *printable* ASCII message encoded in UTF-16BE (without the byte order mark `0xFEFF`). Moreover, you got some witnesses about p such that $p-1$ is divisible by 16 and smaller than 4 times of ciphertext byte length. In your parameter file, you will find the hexadecimal representation of ciphertext c_2. Recover the plaintext in ASCII and provide your answer in `Q2`.

Hint: What happens if we encode an ASCII string with UTF-16BE?

Exercise 3 The Hill Cipher

The Hill cipher was invented by Lester S. Hill in 1929. It is a polygraphic substitution cipher which works as follows:

We use following encoding to transform a character into an integer: "␣ $\rightarrow$ 0", "$a \rightarrow$ 1", "$b \rightarrow$ 2", ..., "$z \rightarrow$ 26"[1]. Let S be an $n \times n$ matrix in $\mathbb{Z}_{27}^{n \times n}$. Then, a message whose

[1]The decoding is straightforward, i.e. inverse of the encoding

length is a multiple of n is encrypted by n characters by multiplying S on the left side. For example, let S be a matrix as follows:

$$S = \begin{bmatrix} 16 & 7 & 18 \\ 20 & 20 & 17 \\ 4 & 5 & 25 \end{bmatrix} \tag{1}$$

Then, the encryption of "abc" will be "cch" since

$$S \cdot m = \begin{bmatrix} 16 & 7 & 18 \\ 20 & 20 & 17 \\ 4 & 5 & 25 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 3 \\ 3 \\ 8 \end{bmatrix} \tag{2}$$

In your parameter file, you will find S_3 decoded with the above encoding where i-th character of S_3 corresponds to the element on $\lfloor i/n \rfloor$-th row and $(i \bmod n)$-th column (assuming the index starts with 0). Decrypt the ciphertext c_3 and write it in `Q3`.

Exercise 4 Bat-Crypt, Diffie-Helman Over The Linear Group

As our favorite hero Batman is trying to find a way to come up with a way to have secure communication with his fellow superheros, he encounters the ElGamal cryptosystem. Being the arrogant superhero he is, he decides to design his own version of the protocol; His motto:"More complicated the group structure, better security!"

Due to his obsession with the movie Matrix, he decides to use the linear group as the discrete logarithm base group, which is defined as follows, where the group operation is usual matrix multiplication:

$$GL(p) = \left\{ \begin{bmatrix} a & b \\ c & d \end{bmatrix}; a, b, c, d \in \mathbb{Z}_p, ad - bc \neq 0 \right\},$$

The group operator being usual matrix multiplication. Being the smartest superhero in the world, he chooses p to be a 256-bit strong prime[2]. The protocol proceeds as follows,

Enc(p, m, g, PK):

1. Alfred selects $s_a \in \mathbb{Z}_{p-1}$ at random and computes $r = g^{s_a}$.
2. Alfred computes $s = \mathsf{PK}^{s_a}$.
3. Alfred sends Batman (r, sm).

Dec(p, c, g, SK):

1. Batman parses $c = (r, C)$.
2. Batman computes $s = r^{\mathsf{SK}}$.
3. Batman obtains the message $m = s^{-1} \times C$

In your parameter file you can find the public parameter p, g, and the public key of Batman PK. Your goal is to find the decryption of the a cipher $c = (r, C)$ found in your parameter file, and write the message (with the python list format) in front of `Q4_plaintext`.

[2]The prime p is called a strong prime if $\frac{p-1}{2}$ is also a prime

$\underline{\mathsf{KeyGen}(p, n, P(x), G(x))}$:
1: Sample y from $\{0, 1, \ldots, n-1\}$
2: $Y(x) \leftarrow G^y(x) \bmod P(x)$
3: **return** $(y, Y(x))$

$\underline{\mathsf{Encrypt}(p, n, P(x), G(x), Y(x), M(x))}$:
1: Sample z from $\{0, 1, \ldots, n-1\}$
2: $Z(x) \leftarrow G^z(x) \bmod P(x)$
3: $V(x) \leftarrow M(x) \cdot Y^z(x) \bmod P(x)$
4: **return** $(Z(x), V(x))$

Figure 1: Key generation and encryption algorithms.

Exercise 5 Polynomial Encryption Scheme

Let p be a large prime and $k > 0$ be an integer. Let $\mathbb{Z}_p[x]$ denote the ring of all polynomials whose coefficients belong to $\mathbb{Z}_p$, i.e. the polynomials are in the form

$$a_0 + a_1 x + a_2 x^2 + \cdots + a_\ell x^\ell$$

where $a_i \in \mathbb{Z}_p$. Let $P(x)$ be a reducible polynomial of degree k. Then we use $\mathcal{R}$ to denote the ring $\mathbb{Z}_p[x]/P(x)$, i.e. all polynomials of degree up to $k-1$. Given two polynomials $Q_1(x), Q_2(x) \in \mathcal{R}$, addition and multiplication operations are defined as below:

$$\begin{aligned} \text{addition:} \quad & (Q_1(x) + Q_2(x)) \bmod P(x) \\ \text{multiplication:} \quad & (Q_1(x) \cdot Q_2(x)) \bmod P(x) \end{aligned}$$

Finally, $\mathcal{R}^*$ denotes the set of units of $\mathcal{R}$, i.e. the subset of elements who have multiplicative inverse. n denotes the order of $\mathcal{R}^*$. Note that $\mathcal{R}^* \neq \mathcal{R} \setminus \{0\}$, as $P(x)$ is reducible. A public-key cryptosystem is defined over $\mathcal{R}^*$, whose key generation and encryption algorithms are described in Figure 1.

In your parameters file, you will find the following values: public parameters p, $P(x), G(x)$ (respectively as `Q5_p`, `Q5_P`, `Q5_G`), a secret key y and a public key $Y(x)$ (respectively as `Q5b_y`, `Q5a_Y`). Moreover, a message $M(x)$, an integer z and a ciphertext $Z(x), V(x)$ are also provided (respectively as `Q5a_M`, `Q5a_z`, `Q5b_Z`, `Q5b_V`. You are expected to:

1. Encrypt $M(x)$ for the public key $Y(x)$ by using the integer z as specified in Figure 1.
2. Figure out the decryption algorithm and then decrypt $Z(x), V(x)$ with the secret key y.

Provide your answers as variables `Q5a_Z`, `Q5a_V`, `Q5b_M` as ASCII strings.
Hint: Square-and-multiply algorithm might come in handy.