Homework 2 - Sage Basics and Prehistoric Crypto

Cryptography and Security 2018

- 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 Q_1,Q_2, Q_3,Q_4, Q_{5a} and Q_{5b}. You can download your **personal** files on `http://lasec.epfl.ch/courses/cs17/hw2/index.php`
- The answers Q_1 and Q_4 should be hex string, the answers Q_2, Q_3 and Q_{5b} should be ASCII string, and the answer Q_{5a} should be a list of bit (in the foramt $[b_1, b_2, \cdots, b_n]$). **Please provide nothing else. This means, we don't want any comment and any 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.
- 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.
- 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.
- For Exercise 4, we recommend you to use **the php page we provide (see the respective exercises for the exact address)**. If you want to automatize your queries, the best option is then to use **the Sage code we provide**. To connect directly to the server, **you have to be inside the EPFL network**[1] (for the php page you don't need to be inside the EPFL network). Use VPN if you are connecting from outside EPFL.

  Alternatively, if you want to do it by hand, use *ncat* (you can download a Windows version on `http://nmap.org/download.html` along with nmap). To send a query QUERY to the server lasecpc28.epfl.ch under port PORT write to the command line

  ```
  echo QUERY | ncat --no-shutdown lasecpc28.epfl.ch PORT
  ```

[1] If you are using Sage Cloud, connecting directly to the server is impossible, as the cloud itself is never inside of EPFL network.

under Windows from a cmd in the same directory as the ncat.exe program

```
echo QUERY | ncat.exe lasecpc28.epfl.ch PORT
```

under MAC OS and some linux distributions, ncat doesn't work as it should and you need to replace it by

```
echo QUERY | nc -i 4 lasecpc28.epfl.ch PORT
```

Here, `--no-shutdown` flag needs to be passed so that local ncat session waits for the server response. If you are using Windows client, you might need to check for a similar flag.

Under Windows, using Putty is also an option. Select "raw" for "connection type" and select "never" for "close windows on exit".

- The homework is due on Moodle on **Wednesday the 24th of October** at 22h00.

Exercise 1 Padding1024

In order to be compatible with most web-based protocols, the apprentice (see Exercise 4) found it convenient to establish a suit of encoding/decoding functions — a library he dubbed "Padding1024". In short, he wanted to have conversion functions among types of hex, int and binascii, which are ubiquitous in SAGE.

In the following, we limit ourselves to integers smaller than 2^{1024} and pad the leftmost bits with zeros to represent every number with exactly 1024 bits. Therefore the hexadecimal encoding is 256 characters and ascii strings are exactly 128 characters.

Let $[n]$ denote the set $\{0, 1, \ldots, n\}$. Let X be an integer such that $X < 2^{1024}$. The binary representation of X is the unique sequence $X_{\mathsf{bin}} = x_{1023}x_{1022}\ldots x_0$ such that $x_i \in \{0, 1\}$ for all $i \in [1023]$ (x_0 represents the rightmost and least-valued bit). It is implicit in our definition that the small values are padded with zeros from the left. Then, the hexadecimal representation of X is the unique sequence $X_{\mathsf{hex}} = h_{255}h_{254}\ldots h_0$ such that $h_i \in \{0, 1, \ldots, e, f\}$ for all $i \in [255]$ and $h_j = \sum_{i=0}^{3} 2^i \cdot x_{i+4\cdot j}$ in base-16. Similarly, X_{byte} is the sequence $b_{127}b_{126}\ldots b_0$ such that each b_i is a byte (or ascii-extended character), and $b_i = \sum_{i=0}^{7} 2^i \cdot x_{i+8\cdot j}$ holds for all $i \in [127]$.

Hint: In SAGE, you will find it convenient to use X_{byte} encoding, as input and output of `hashlib.sha256` is byte string.

In this exercise, we ask you to *acclimatize* yourself with the errands of encoding and encourage you to implement Padding1024 to get a better understanding of SAGE[2]. You are given a strong prime p (i.e. $(p-1)/2$ is also prime) and a subgroup of $\mathbb{Z}_p^*$ generated by g. Your parameter file contains p_{hex} (given as p_1) and g_{hex} (given as g_1), i.e. the values are hex encoded with Padding1024. Find the order of the subgroup generated by g, and provide in $Q1$ encoded with Padding1024. Your answer must be exactly 256-hexadecimal-character string wrapped in double or single quotation marks, just as the ones in parameter file.

[2]Feel free to use any library function in your favorite programming language, or wrap any standard function, all is allowed. Keep your library around for the following exercises and homeworks.

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. Actually, he found that he can generate a random key sequence from a secret seed. In CA-Vernam cipher, there exists a secret seed *seed* of arbitrary length which is only shared between the sender and the receiver instead of a shared key whose length is equal to the length of message. Let H be a hash function which maps any string to h byte string, and $m = m_0||\cdots||m_{n-1}$ be an n byte ASCII encoded message to encrypt. Then, the encryption of m is $c = c_0||\cdots||c_{n-1}$ where $c_i = m_i \oplus k_i$ for $i \in \mathbb{Z}_n$ and k_i is defined as

$$k_{ti}||\cdots||k_{t(i+1)-1} = \begin{cases} \mathsf{trunc}_t(H(seed)), & \text{if } i = 0 \\ \mathsf{trunc}_t(H(k_0||\cdots||k_{ti-1}||m_0||\cdots||m_{ti-1})), & \text{otherwise} \end{cases}$$

where $t = \gcd(h, n)$ and $\mathsf{trunc}_i(x)$ takes leftmost i bytes of x. 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, and you know that SHA256 was used as H for the encryption. In your parameter file, you will find the hexadecimal representation of ciphertext c_2. Recover the plaintext and provide your answer in Q_2.

Hint: the plaintext is printable.

Exercise 3 The Playfair Cipher

The playfair cipher was invented by Charles Wheatstone in 1854. It is a digram substitution cipher, i.e. substitution by two letters. It works as follows:

Let $\mathcal{M}$ be a set of 25 characters and $m = m_0 m_1 \cdots m_{2n-1} \in \mathcal{M}^{2n}$ be a string to be encrypted where $m_{2i} \neq m_{2i+1}$ holds for any $i \in \mathbb{Z}_n$. Then, given a 5×5 matrix K whose elements are distinct characters in $\mathcal{M}$, the encryption works as follows. For $i \in \mathbb{Z}_n$, we firstly find the location of m_{2i} and m_{2i+1} on K. There are 3 cases:

- If they are on the same column, we take the element on their next row[3] as its encryption.
- If they are on the same row, we take the element on their next column[4] as its encryption.
- Otherwise, we draw a rectangle on K such that m_{2i} and m_{2i+1} are vertices of the rectangle. Then, we substitute them by the element of the vertex which is on the same row.

Moreover, we encode the key K as a string by iterating from top-left to bottom-right by row. For example, let K be a table as follows.

[3]Next row of the last row is the first row.

[4]Next column of the last column is the first column

a	b	c	d	e
f	g	h	i	k
l	m	n	o	p
␣	r	s	t	u
v	w	x	y	z

Then, K can be encoded as "abcdefghiklmnop rstuvwxyz" and the encryption of "seed" is "ucae". In your parameter file, you will find the key[5] K_3 and the ciphertext c_3. Recover the plaintext and provide your answer in Q_2.

Exercise 4 The Password-based Key Exchange

After having a perfect grade in his first quiz, the apprentice started to feel quite confident that he comprehended the basics of security, therefore decided to implement a better version of password-based login system on top of Diffie-Hellman key exchange.

As much as being pragmatic and enthusiastic, the apprentice was also realist so he made the design choices that truly reflect the imperfections of the reality. He decided that the password domain should be small, i.e. it is feasible to enumerate and exhaust all passwords. Namely, each user in the system picks a pwd uniformly from $\mathbb{Z}_{10^5}$ and transmits it to the server via authenticated and confidential out-of-band channel, assuming that things went smoothly at sign up time. He believed that he can develop a key-exchange protocol where the server accepts and derives necessary key material only if the interacting user knows the password, and otherwise rejects the communication altogether and halts.

His protocol design is fully depicted in Figure 1. (p, k, g) tuple is the public parameter (given as (p_4, k_4, g_4) in your parameter file) such that g generates a subgroup of $\mathbb{Z}_p^*$ with order k. As you have seen in the lecture, it is believed that the computational Diffie-Hellman problem is hard for a sufficiently large prime-order subgroup of $\mathbb{Z}_p^*$. However, the apprentice was sloppy and hasty, so he did not properly test if k is a prime.

His design intuition was to hide the generator of Diffie-Hellman by using the password. Essentially, both parties use pwd to pick a *hidden* generator $\hat{g} = g^{\mathsf{pwd}} \mod p$ and perform Diffie-Hellman key exchange as usual. The user sends a confirmation value ACK that depends on the derived key K and the protocol transcript (X, Y) so as to confirm that the user was able to deduce the value of K, therefore he knows pwd.

Your goal is to find the password by interacting with the user oracle whose password is fixed. Recover pwd and provide your answer in Q_4 with Padding1024. I.e. the password should be extended to exactly 256 hexadecimal characters, and wrapped in string quotation marks.

Hint: The apprentice's choice of *fake* prime k was so bad that there is a small prime q ($|q| < 30$, i.e. $q < 2^{30}$) dividing k. You might find `factor` function of SAGE handy.

You have access to **the oracle which simulates the user** at web interface `http://lasec.epfl.ch/courses/cs18/hw2/useroracle.php`. By using this web interface, you can interact with the user, imitating a server who knows the password. Alternatively, you can connect to the server lasecpc28.epfl.ch on port 5555 using Sage or ncat to issue queries directly (see instructions above). Your query should consist of two arguments: your SCIPER number and the hex encoded value of your query X_{hex}. An example from shell would be:

[5] It consists of 24 lower case alphabet characters except j and q, and the space.

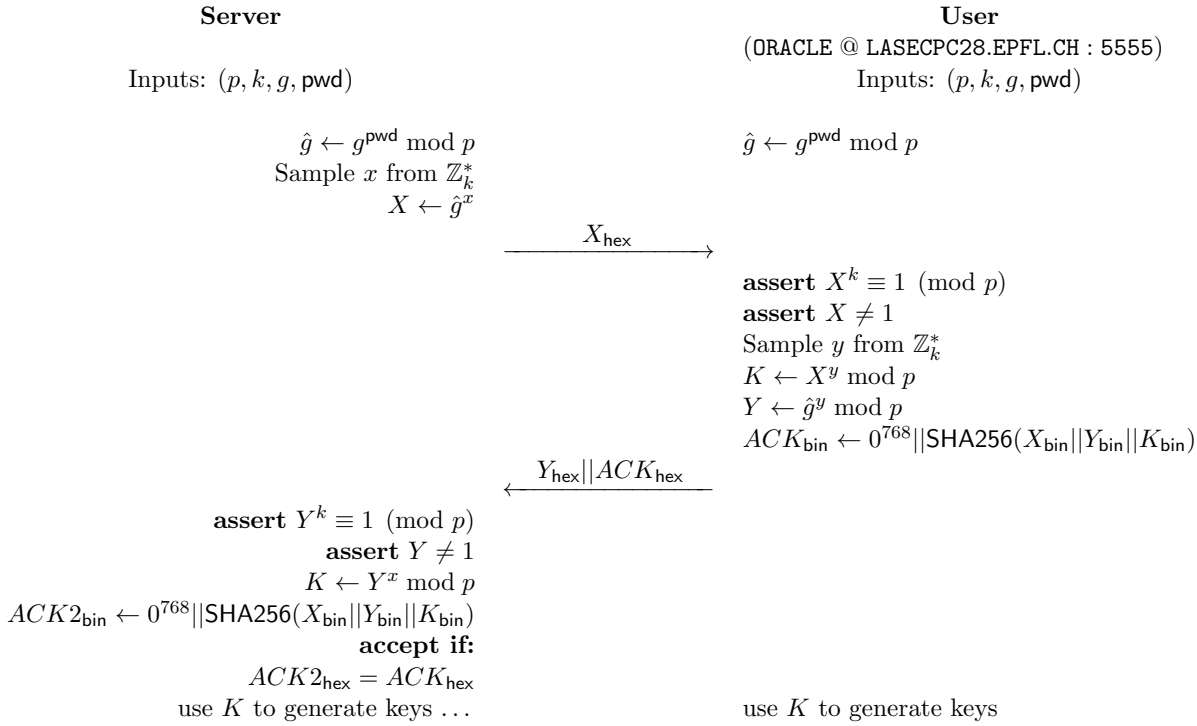

Figure 1: The notation follows the one introduced with Padding1024 (see Exercise 1): X_{bin} and X_{hex} represent the binary and hexadecimal representations of integer X. $||$ operator is string concatenation. Each invocation of $\mathsf{SHA256}$ takes exactly $1024 \cdot 3 = 3096$ bits of input and returns 256 bits, i.e. $\{0,1\}^{3072} \mapsto \{0,1\}^{256}$. However, for consistency with Padding1024, the output of $\mathsf{SHA256}$ is subsequently left-padded with 0^{768}. All transmitted elements ACK_{hex}, X_{hex}, Y_{hex} are of 256 hexadecimal characters. The user's message $Y_{\mathsf{hex}}||ACK_{\mathsf{hex}}$ is string concatenation of Y_{hex} (on the leftmost side) and ACK_{hex}.

```
echo 123456 0000a0979c213ebd7f7278f3454d01d7fef8da53a21cbae9da159dc9283a7a178
a9f1fa65fbcd271a204dc18481612049d0986717e1b8289307f2bb71f8cc44fc88493b6705469
e17b1a5d342a1f1d7cc623c4d4685d4e94be6a01de82e23c44108d18afb70fcd1a5ec94bbfdfd
fd4aa46a518cb779ab4a876d4b7b2d52e5c6e | ncat --no-shutdown lasecpc28.epfl.ch
5555
```

which returns the concatenated value of $Y_{\mathsf{hex}}||ACK_{\mathsf{hex}}$ which is 512 characters long as below:

```
00014f54b2295513afff5449e1ebfea058d27ecdfc0061ad250c153ce652b5631272718fbdb66
6ac104e28cdd622515cd4ce6fc8c48411492c5d03967792343ec0ba7c8d335fe64a03ea1a6494
0ad61107781a8dc29163a7731261cc4a43eac7172e3dac10e117df8695bd4f9bc68d2696eea35
3785c3697d5b390cb5aae83cb0000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000b628ad57cec6c7
184f6d15a71451fc277eda89dec06735ed560fba541894ebee
```

You do not need to use string quotation character " in your queries, and you will not receive any in oracle responses.

Exercise 5 Badly Used ElGamal Cryptosystem

Our crypto-apprentice discovered (while reading the rest of the second chapter of the Cryptography & Security course) that ElGamal cryptosystem[6] gives a way to encrypt his top-secret messages randomly. Indeed, he decided to use it by encrypting his strings of messages character by character since the same characters will not be encrypted to the same ciphertext due to the randomness. The apprentice noticed that his messages only consist of alphabetical characters and space. Thus, he found a special encoding for each character without paying attention to the details of the ElGamal cryptosystem!

Our naive apprentice used the following encoding of characters: "$\llcorner \to 0$", "$a \to 1$", "$b \to 2$", ..., "$z \to 26$" [7]. Furthermore, he noticed that using SSH2 parameters (p, q, g) would be a great idea, where p is a prime, q is a prime order of a subgroup of a group whose order is $p-1$, and g is the generator of this subgroup of order q. More concretely,

$$
\begin{aligned}
p &= 2^{1024} - 2^{960} - 1 + 2^{64}\lfloor (2^{894}\pi + 129093) \rfloor \\
q &= \frac{p-1}{2} \\
g &= 2
\end{aligned}
$$

In ElGamal cryptosystem, the messages must be from the subgroup of order q generated by g.

You have intercepted the ciphertext, and are determined to recover the plaintext without the key. You make a (very good) guess, that the message only contains "normal" English words, which are all listed in the file `dictionary.txt` provided by us, and spaces. To warm up, we would like you to work with a small set of ciphertexts in order to find out which of the characters are encrypted "without" respecting the ElGamal cryptosystem. In your parameter file, you will find a list of pairs of integers C_5 and a list of five integers idx_5. Namely, $C_5 = [ct_0, ct_1, \ldots, ct_k]$ is the list of pairs of integers (i.e. ciphertexts) that are generated from

[6] check the slide 219 of the lecture for the details.

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

some subset of encoded characters without paying attention if the encoding is in the subgroup, and $idx_5 = [i_0, i_1, i_2, i_3, i_4]$ is the list of indexes where $i_j \in \{0, 1, \ldots, k\}$. First of all, generate another list $Q_{5a} = [b_0, b_1, b_2, b_3, b_4]$ where $b_j = 0$ if the **decryption** of ct_{i_j} is **not** in the message space and $b_j = 1$ otherwise, $\forall j \in \{0, 1, 2, 3, 4\}$. Write it down in your answer file. Secondly, given C_5 that is the encryption of a message, recover the original plaintext Q_{5b} with the help of the dictionary (use only the plaintext words that exist in the given dictionary). Write it in your answer file.