Problems for week 2, Cryptography Course - TDA 352/DIT 250

General remarks on problems for the weekly problem session: Exercises will be classified in four different levels:

1. **Easy**: the exercise will require low numerical computations or it can be just a way to look back at the content of the lecture. Exercises of this level should easily be done with just *pen and paper* and are **important to pass** the exam.
2. **Medium**: the exercises will require some time to do (from 5 to 15 minutes each). Maybe a separate paper for some computation is needed! You need to study a bit to answer the questions. These exercises also **may appear** in the exam.
3. **Hard**: the exercises will require you to spend a lot of time doing numerical computations (and we highly recommend using a PC) **or** the questions are real challenge to see if you understood the course in depth. Some of these exercises **may appear** in the exam.
4. **Think**: problems that aim to using your imagination. You are invited to discussion with your colleagues/friends/family and find your best solutions. Generally, the exercises of this level do not take a lot of time in writing the solutions but they will let you think/discuss for (maybe) 30/40 minutes.

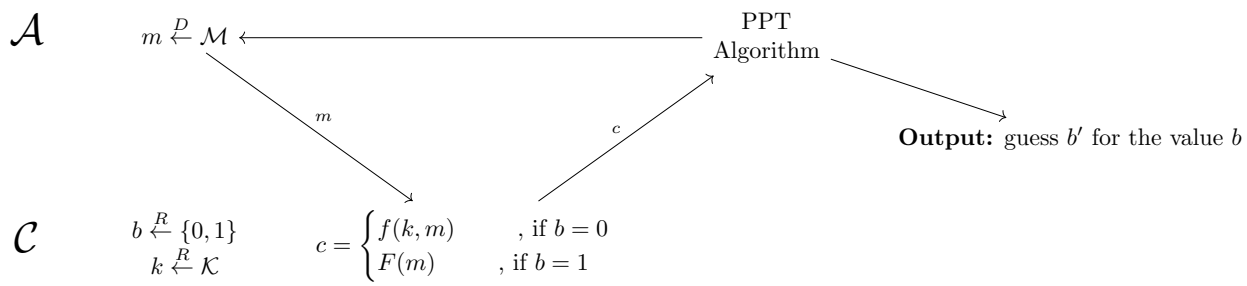
In this weekly exercise sheet: you will use PRG, PRF, PRP, block ciphers and their mode of operation.
Completing the ex. sheet: you will be able to prove basic secure PRF, get familiar with the definitions, own a personal block cipher and some good ideas on block ciphers.

Easy

---

1. **Define** secure PRG, PRF, PRP.
   What are the difference between them? Do we have some insiemistic inclusion, i.e., one is a subset of another?
2. **Describe** ECB and CBC mode of operation (encryption and decryption).
3. Let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a secure PRF. (where the key space, input space, and output space are all $\{0,1\}^n$) and let $n = 128$.
   Prove if the following function defined using f are secure PRFs or not:
   (a) $f'(k,x) = \begin{cases} f(k,x) & \text{if } x \neq 0^n \\ k & \text{otherwise} \end{cases}$
   (b) $f'((k_1,k_2),x) = f(k_1,x) \oplus f(k_2,x)$
   (c) $f'(k,x) = k \oplus x$
   (d) $f'(k,x) = f(k, x \oplus 1^n)$
   (e) $f'(k_2,x) = f(0^n,x) \| f(k_2,x)$

   Remark: In order to prove that a function f is a secure PRF, an attacker should never win at this game, where F is a random function from $\{0,1\}^n \to \{0,1\}^n$

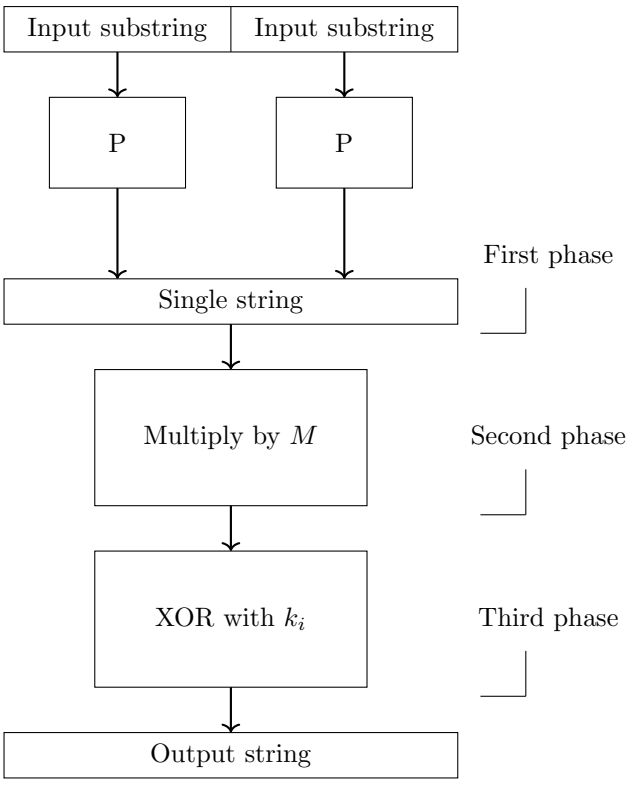


Observe that the attacker can ask as many queries as he desire.

4. Since in the previous exercise sheet, we constructed a stream cipher (using a construction of a) single round of a Feistel block cipher, this week we will create a block cipher with the **substitution-translation scheme** (in mathematical papers sometimes called translation based block cipher).
These ciphers have a unique and well defined structure for the ith-round which can be subdivided in three phases:

   - First phase:
     - at the beginning, the input of the i-th round is subdivided in substrings[1]
     - the substitution value is computed for every substring
     - all the different substituted value are concatenated in a single string
   - Second phase:
     - the single string is multiplied by an invertible matrix M witch has elements entries in $\mathbb{Z}_2$
   - Third phase:
     - the result of the multiplication is xor'ed with k_i, the i-th round key
   - the result of the xor is the output of the round

   The diagram below depicts the the before mentioned subdivision:



   In order to construct the block cipher, we need a random substitution function (P in the diagram) on the 3-bit strings, a matrix M and a key scheduler.
   To keep the construction as easy as possible, the key-scheduler just returns the same key k for all the different rounds. This means that you choose the secret key k and set all the round keys equal to k.
   As invertible matrix, you can use the M, with inverse M^{-1}, defined as:

$$M = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix} \qquad M^{-1} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

   You are free to use the substitution P that you used last week or creating it to a new one! An example, this is a possible substitution function P is the following:

[1] The number of substrings depends on the substitution function P.

x	0	1	2	3	4	5	6	7
x in bit	000	001	010	011	100	101	110	111
P(x)	5	3	2	6	0	1	7	4
P(x) in bit	101	011	010	110	000	001	111	100

In a more mathematical way, you can see the round as the function

$$r(k,x) : \mathbb{Z}_2^6 \times \mathbb{Z}_2^6 \to \mathbb{Z}_2^6 \qquad r(k,(x_1\|x_2)) = k \oplus M(P(x_1)\|P(x_2))$$

The final construction step is iterating the round a fixed number of time!

Task: choose the number of round and build your block cipher! Try to encrypt/decrypt some messages.
If you want to, you can implement it in a programming language and then exchange or check it with your friends!

Medium

---

5. **Prove** that ECB is not semantic secure.

6. An important assumption widely believed by cryptographers is that there exists different problem that are computationally hard to solve. For example factorizing an integer or finding some particular solution for mathematical problems like the discrete logarithm (or DLog).
   For this reason, we measure the **computational complexity** as *how many computations we need to do to get the result.* The computational complexity is a characteristic that can be computed by looking at **how** algorithms work.

   For example, let's assume that doing the xor between two bit string is really, really hard and we want to see *how much* hard it is and so we consider the algorithm for the xor operation for two n-bit strings x, y:

   - Start from the first position, i.e. $i = 1$ (here i is the index used in the algorithm)
   - Take the ith-bit for x as x_i (similarly for y, obtaining y_i). Assume[2] that the number of computation needed is n
   - Compute $x_i \oplus y_i$ and save it in z_i. Assume that the cost of this operation is n^3
   - If we are at the end of the string, we have $z = x \oplus y$.
     Otherwise increase i by 1 and start from the second point

   As a Python-like pseudo-code:

```
# Input : x, y
# Output: x xor y

def xor(x,y):
  n = length(x)

  # The solution is z
  # Initialized as a 0 array of length n
  z = [0,0,...,0]

  for i in range(1,n):
    z[i] = x[i] xor y[i]

  return z
```

   This algorithm will take x, y of length n and output $x \oplus y = z$.

   considering the whole algorithm, we want to know the *total cost* of executing it. So we fix the inputs as x, y of length n and observe that we are repeating n times (one for every position) the *compute* z_i.

[2] **we** basically decided **how** we extract the ith bit. It may exists a better algorithm!

So we are doing n times n^3 operation.
We so get that n^4 will be our computational complexity and when we fix n to be a number, we will exactly know how many operation we will have to know.

Questions:

- Suppose that $n = 30$. We may compute the xor by hand and so we have a **computational capability** of 5 operations per second, i.e. we are able to do 5 operations in one second, 10 op. in 2 seconds and so on.
  How long does it take us to do the whole computation (i.e. computing the xor)?
- Now suppose that we are really tired after the last xor computation and now can do 1 operation per second and researchers found **the** best algorithm that has computational complexity of n^2. How much time we will spend in this case?

7. **Define** what is a block cipher. **Describe** how DES and AES are constructed and in what they differ.

Hard

8. **Question:** Why is DES breakable?
   Hint: *try to search on the Internet some different reasons.*

9. **Prove** that CBC is not secure for chosen plaintext attack (CPA).

10. **Define and describe** the CPA security game

11. The Data Encryption Standard (DES) has been a predominant symmetric-key algorithm for the encryption of electronic data. In 1976 DES was approved as a federal standard in the US. However, due to *short* key size (56-bit) DES is now considered to be insecure for many applications.

    An improvement, proposed by Rivest, is DESX. DESX uses a key pair (k_1, k_2), where k_1 is 56 bits and k_2 64 bits. The encryption of a one-block message m is defined as $DESX_{(k_1,k_2)}(m) = \text{Enc}_{DES}(k_1, (m \oplus k_2)) \oplus k_2$.
    Explain how DESX decryption works. Consider the (insecure) variant without inner xor-ing:

    $$DESX'_{(k_1,k_2)}(m) = \text{Enc}_{DES}(k_1, m) \oplus k_2$$

    Find an attack against $DESX'$ that is better than brute force.

Think

12. **Question:** Suppose we have a message m and its AES encryption, i.e. $c = \text{AES}(k, m)$ where k denotes the secret key. From all the information we have, can we consider AES as a perfect/secret cipher?

13. **Question:** Why in a block cipher we have a PRP and not a RP, i.e., simply a random permutation?

14. **Question:** List the cryptosystems that we can build from a single substitution function and explain the differences.