Problems for week 1, 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 some historical ciphers, the OTP, the definition of semantic security and some combinatorial problems.
Completing the ex. sheet: you will be able to prove semantic security, get familiar with the definitions, own a personal stream cipher and some ideas on *why* randomness is important in Cryptography.

Easy

---

1. Alice wants to lend her credit card to Bob. To do this, she needs to tell Bob her PIN: a 5-digit secret combination. Alice, who followed the first lectures of the Cryptography course, chooses to encrypt her PIN in this way:
   - she chooses a substitution function on the single digits $\{0, 1, \ldots, 9\}$ (or more accurately $\mathbb{Z}_{10}$), that is a 1 to 1 function from $\{0, 1, \ldots, 9\}$ (or more accurately $\mathbb{Z}_{10}$) to $\{0, 1, \ldots, 9\}$ (or more accurately $\mathbb{Z}_{10}$). This means that one input digit will correspond to a single output digit. For example a possible substitution:

Original	0	1	2	3	4	5	6	7	8	9
Substitution	5	3	2	9	0	1	8	4	7	6

     To encryp a digit then means to look at the number in the first row in the table above and rewrite it with the number in the second row.
   - Alice then encrypt every number of her PIN and then gives the credit card, the encrypted PIN and the substitution function to Bob.
   - Bob swaps the rows of the substitution and obtains:

Substitution	5	3	2	9	0	1	8	4	7	6
Original	0	1	2	3	4	5	6	7	8	9

     Bob can now decrypt Alice's PIN by looking at the table.

   Assume now that Eve manages to steal the credit card and the encrypted PIN which is

   53287

   Questions

   (a) Can Eve found the original PIN? Why?

   (b) Suppose that the substitution used by Alice is

Original	0	1	2	3	4	5	6	7	8	9
Substitution	5	3	2	9	0	1	8	4	7	6

Can we now decrypt Alice's PIN?

2. Alice, after her first attempt with the substitution method, wants to change the way she encrypts her numerical informations. She construct a Vigenère cipher that encrypts numbers in the following way:

   - Alice chooses a secret number with 3-digit (for example 827) and gives this number to Bob.
   - Alice takes the number to encrypt, for example the Dallas ZIP-code 75216, and writes the table

ZIP-code	7	5	2	1	6	
+ Secret	8	2	7	8	2	7
Sum	15	7	9	9	8	

     and since she wants to have a single encrypted digit for every digit of the ZIP-code (the number 15 has two digits), she keeps just the reduction *modulo 10*, so she keeps the digit 5, i.e. $15 \bmod 10 \equiv 5$ (essentially Alice is doing operations in $\mathbb{Z}_{10}$).
   - Alice sends the encrypted ZIP-code 57998 to Bob
   - Now Bob decrypts in the following way:

Encrypted ZIP-code	5	7	9	9	8	
- Secret	8	2	7	8	2	7
Difference	-3	5	2	1	6	

     and every time he finds a negative number, Bob *"adds* 10*"* until the number is positive. So, e.g., -3 became 7. He mathematically does the *reduction modulo* 10.
   - Bob now has the original number 75216

   Questions and tasks

   (a) Suppose you are Alice, and the secret number is 827.
   Encrypt these GPS-coordinate seen as numbers

   27751775 98069974 27551292 99451882

   and send them to Bob!

   (b) Now you are playing Bob's role. Decrypt the messages you got from Alice and, since you know the original message, see if the decryption is correct.

3. **Question:** Explain why stream ciphers cannot have perfect secrecy?
   Hint: *you should compare the OTP cipher and the stream cipher and look for the differences between them.*

4. **Question:** Define Perfect Secrecy in a formal way and try to informally explain the meaning of the definition.

5. **Task:** Describe in a formal and complete way, the OTP encryption scheme.

Medium

---

6. Define, in a formal way, Semantic Security.
   Now consider (E, D) be a (one-time) semantically secure cipher where the message and ciphertext are strings of $0, 1$.
   Which of the following encryption schemes are semantically secure?

   (a) $E'(k, m) = 0 \| E(k, m)$
   (i.e. prepend 0 to the ciphertext)

   (b) $E'(k, m) = E(k, m) \| k$

   (c) $E'((k, k'), m) = E(k, m) \| E(k', m)$
   (i.e. the key is splitted in two part, and then the two encryptions are concatenated together)

(d) $E'(k,m) = E(0,m)$

(e) $E'(k,m) = E(k,m)\|\text{LSB}(m)$

Remark: with the symbol $\|$ we intend the string concatenation, i.e. $111\|000 = 111000$. With the symbol LSB we intend the *Least Significant Bit* of the string.

7. To prove perfect secrecy, we want to count the dimension of the key space $\mathcal{K}$ containing bit-strings of length 10.
Find, and explain how you get the dimensions of the following key-space and the if randomly chosen keys have the same probability uniform distribution, i.e. finding the probability distribution of the key-space:

   Remark: sometimes we will *flip a coin*. The result of the coin will be encoded in 1 if the coin returns *head* (or H), 0 if the coin returns *tail* (or T).
   For example: if we flip the coin 4 times and get H,H,T,H (in this order), we can see the result as 1101.

   (a) All the possible keys obtained by launching a coin 10 times

   (b) Launch 5 times a coin. Then concatenate the result with itself to get the key

   (c) Take an integer number from 0 to 1000. The binary representation of the number is the key (fill with 0s the empty spaces)

   (d) Take an integer number from 0 to 2000. The binary representation of the number is the key (fill with 0s the empty spaces or eliminate the most significant bit if the string is longer than 10 bits)

8. In practice, to build PRG, cryptographers exploit existing cryptographical *structures* [1] to build new *structures* such as protocols, different hash functions, cipher with special properties and many, many more.
   This is possible since Cryptography is build over mathematical/algorithmic primitives that we can see as *elementary bricks*. Then, from the bricks, cryptographers build *walls* as more complex cryptographic schemes. From walls, we build *houses* and so on...

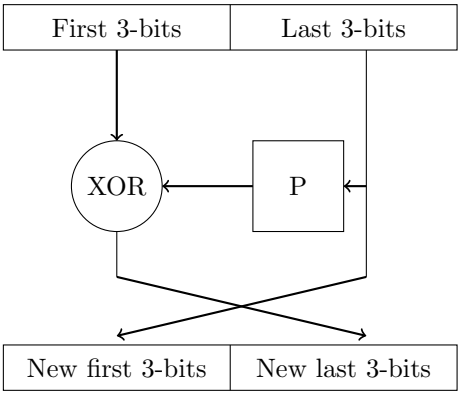
   In this exercise, you will build your personal stream cipher (the *house*) and to do it, you will create a PRG (the *wall*) using just a substitution function (the *brick)*! We will use a modified (and simplified) construction that A5/3[2] uses.
   A5/3 is a stream cipher used in 3G communication: the Internet on your smartphone!

   For the construction, you only need to choose your personal substitution, that we will call P, of the digits from 0 to 7.
   Formally you are choosing a $1-1$ (bijective) function $P : \mathbb{Z}_8 \to \mathbb{Z}_8$ where $\mathbb{Z}_8 = \{0, 1, \ldots, 6, 7\}$.
   We will use the representation of that digits as 3 bit strings.
   For example, consider this substitution 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

   Let's consider this particular construction that we will use to generate the pseudorandom bits.

[1] such as PRG, stream cipher, substitution function and many more that you will see in the other lectures!
[2] Wikipedia link to the general description of A5/3.



We will call this strange operation *round*.

To use the stream cipher you have to:

(a) Choose a secret key K that will be a 6-bit string. Store the value of K in a new variable x

(b) Input x into the *round*

(c) Compute the output

(d) The second most significant bit, is our *random generated* bit

(e) If you want a new pseudorandom bit, start from the point (8b) changing the variable x with the output you computed

(f) When your generated string is long enough, you can use the generated string as a OTP-key, i.e. you XOR the generated string with the message to encrypt

For security reasons, the first bit of the string generated, will **always** be dropped
Now you have build your personal stream cipher.

An example, using the previous defined substitution P and secret key $K = 000111$, is provided to explain the computational steps:

$K = 000\|111$ **Round 1**

(a) $x = 000\|111$

(b) $P(111) = 100$

(c) 000 xor 100 = 100

(d) **Output:** 111‖100 and the random bit is 1

Round 2

(a) $x = 111\|100$

(b) $P(100) = 000$

(c) 111 xor 000 = 111

(d) **Output:** 100‖111 and the random bit is 0

So after 2 rounds, we get the random string 10.
We drop the first bit (1) and get 0 as the first bit generated by the PRG that we can use to encrypt a single bit in our stream cipher!

Task: build your own stream cipher and try to generate some bit-strings of pseudorandomness!

Information: this particular construction will be described and used in the future, both in a lecture and exercise session! Try to understand *how* it works!

9. Since, sadly, a mathematical definition hides (sometimes) infeasible problems, the real world cryptographers test stream ciphers fixing some predefined seeds, they generate the pseudorandom string s from the output of the stream cipher and then run some fixed statistical (and well studied) tests on the string.

   Let s be a fixed length string of bit 0,1 that is generated by your personal stream cipher. Suppose we fix the string's length to 10.

   We want to test if our stream cipher is a secure PRG.

   The testing seeds are

   $$010101 \qquad 101010 \qquad 111000$$

   and our statistical test are

   $$\mathcal{T}_1(x) = \begin{cases} 0 & \text{if the number of } 0 - \text{ to the number of } 1 \text{ is } < 4 \\ 1 & \text{otherwise} \end{cases}$$

   $$\mathcal{T}_2(x) = \begin{cases} 0 & \text{if the string does not contain the substring 00000 or 11111} \\ 1 & \text{otherwise} \end{cases}$$

   Then, we will fill the table with the result of the tests

	010101	101010	111000
$\mathcal{T}_1$			
$\mathcal{T}_2$			

   We say that *"the stream cipher passes the statistical test"* if the tests always output 0.

   Questions and tasks

   (a) Test your stream cipher and check its performance with your colleagues!

   (b) Not passing some tests is **not** a sign of *non*-randomness. Why?

10. **Question:** Does the OTP guarantee perfect secrecy? Is it semantically secure? Prove it!

Hard

10. Prove that a secure PRG is unpredictable.
    Hint: *prove the contrapositive, i.e. that a predictable PRG is not secure.*

11. Hill's cipher is a particular cipher that has a matrix as a secret key and encrypts (respectively decrypts) by multiplying the message with that secret matrix (respectively it's inverse).
    For example: consider that the messages we want to send are just the numbers in $0, 1, 2, \ldots, 9, 10$ (the *reduction modulo 11*).
    Consider this two matrices

    $$E = \begin{pmatrix} 1 & 3 & 0 \\ 3 & 1 & 2 \\ 0 & 2 & 1 \end{pmatrix} \qquad D = \begin{pmatrix} 3 & 3 & 5 \\ 3 & 10 & 2 \\ 5 & 2 & 8 \end{pmatrix}$$

    To encrypt a message $m = (m1, m2, m3)$ you will encrypt computing $E \cdot m$ and decrypting with $D \cdot m$, using the matrix multiplication and remembering to always reduce *mod 11*.
    As a computational example: the encryption of the message $m = (4, 0, 2)$

    $$E \cdot m = \begin{pmatrix} 1 & 3 & 0 \\ 3 & 1 & 2 \\ 0 & 2 & 1 \end{pmatrix} \cdot \begin{pmatrix} 4 \\ 0 \\ 2 \end{pmatrix} = \begin{pmatrix} 4 \cdot 1 + 0 \cdot 3 + 2 \cdot 0 \\ 4 \cdot 3 + 0 \cdot 1 + 2 \cdot 2 \\ 4 \cdot 0 + 0 \cdot 2 + 2 \cdot 1 \end{pmatrix} = \begin{pmatrix} 4 \\ 16 \\ 2 \end{pmatrix} = \begin{pmatrix} 4 \\ 5 \\ 2 \end{pmatrix}$$

    the encrypted message is $(4, 5, 2)$.

- **Task:** Encrypt the following messages:
  (a) $m = (1, 1, 1)$
  (b) $m = (7, 10, 3)$
  (c) $m = (5, 2, 7)$
- **Task:** Decrypt the following messages:
  (a) $m = (1, 1, 1)$
  (b) $m = (7, 10, 3)$
  (c) $m = (5, 2, 7)$

12. **Question:** What is the role of randomness in cryptography?

Think

13. An attack where the adversary is even more powerful is the *chosen plaintext attack*, where the adversary can submit as many plaintexts of her choice and obtain the corresponding encryptions. Suppose that the adversary knows exactly how the encryption scheme works but not the key.
Try to find a practical attack (or just an idea on *how* you should do it) to obtain the key for
    (a) the substitution cipher of Exercise 1
    (b) the Vigenére cipher of Exercise 2
    (c) the Hill's Cipher of Exercise 11

14. *(Ex. 2.13 of Katz-Lindell book "Introduction to Modern Cryptography")*: Let us consider this variation on Shannon's Perfect Secrecy definition.
We have the message space $\mathcal{M}$, the key-space $\mathcal{K}$ and ciphertext space $\mathcal{C}$. Consider M_1 and M_2 as the random variables denoting the first and second message that we want to encrypt with the same key k, respectively.
We so choose randomly a single key $k \leftarrow \mathcal{K}$, and encrypt two messages m_1 and m_2 in $c_1 \leftarrow \text{Enc}_k(m_1)$ and $c_2 \leftarrow \text{Enc}_k(m_2)$; this induces a probability distribution over the ciphertexts and we let C_1, C_2 be the corresponding random variables.

    Define the **perfect secrecy for two messages** as an encryption scheme $(\text{Gen}, \text{Enc}, \text{Dec})$ if for all the random distribution of the two messages M_1 and M_2 and ciphertexts C_1, C_2 with a fixed key k such that for all ciphertexts $c_1, c_2 \in \mathcal{C}$ with $\Pr(C_1 = c_1 \wedge C_2 = c_2) > 0$:

$$\Pr(M_1 = m_1 \wedge M_2 = m_2 | C_1 = c_1 \wedge C_2 = c_2) = \Pr(M_1 = m_1 \wedge M_2 = m_2)$$

    Prove that **no** encryption scheme can satisfy this definition.

15. **Question:** Why is it important, in the Shannon Perfect Secrecy, that $\text{len}(m_1) = \text{len}(m_2)$ (the length of two messages is the same)?