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

In this weekly exercise sheet: you will use PRG, PRF, PRP, block ciphers and their operational mode.
Completing the exercise 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. **Definition**: A function $\mathcal{G} : \{0,1\}^l \to \{0,1\}^n$ with $l \ll n$, is a *secure PseudoRandom Generator* if for every efficient statistical test $\mathcal{T}$, it holds that

$$|\Pr(\mathcal{T}(\mathcal{G}(u)) = 1) - \Pr(\mathcal{T}(\mathcal{G}(u)) = 0)| \text{ is negligible}$$

for every $u \overset{R}{\leftarrow} \{0,1\}^n$ picked uniformly at random and $\mathcal{T}(x) = \begin{cases} 1 & \text{if } \mathcal{G}\text{'s output considered not random} \\ 0 & \text{otherwise} \end{cases}$

Definition: A function $\mathcal{F} : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$ is a *Pseudo Random Function* if the function $\mathcal{F}_k = \mathcal{F}(k, \cdot)$, for a uniformly random selected key $k \overset{R}{\leftarrow} \mathcal{K}$ is *indistinguishable* from a function chosen uniformly ar random from the set of all possible functions from $\mathcal{M}$ to $\mathcal{C}$.

Definition: A PRF $\mathcal{F} : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$ is a *Pseudo Random Permutation* if all the following statements hold:

(a) $\mathcal{M} = \mathcal{C}$

(b) the function $\mathcal{F}(k, m)$ is a one-to-one (i.e., bijective, invertible)

(c) there exist an efficient, deterministic algorithm to compute $\mathcal{F}(k, m) = E(k, m)$ for any message $m \in \mathcal{M}$

(d) there exist an efficient, deterministic algorithm to compute $\mathcal{F}^{-1}(k, m) = D(k, m)$ for any message $m \in \mathcal{M}$ (recall that by point (a) $\mathcal{M} = \mathcal{C}$, so in this case the message is also a ciphertext)

From the definition, we derive that a PRP is a special case of PRF where $\mathcal{M} = \mathcal{C}$. Thus set of PRPs is contained in the set of PRFs.
PRFs need a key and, by definition, fixing a key the output of the PRF is a random variable indistinguishable from a real random variable. This means that a $\mathcal{F}_k$ will pass every statistical test. So fixing a key gives us a secure PRG. But a PRG can not be a PRP because in the secure PRG definition, we need that $l \ll n$ and for PRP we have $l = n$.

2. Let (E, D) be a block cipher.
ECB: in ECB, the data M is divided into blocks $\{m_i\}$ and each block is separately encrypted using E and the key k. If we denote with c_i the i-th encrypted block, we have that

$$c_i = E(k, m_i) \qquad m_i = D(k, c_i)$$

CBC: in CBC, the data M is divided in blocks $\{m_i\}$, a random string initialization vector IV is

chosen and each block is encrypted with E and a key k. The IV is placed at the beginning of the ciphertext. If we denote with c_i the i-th encrypted block, we have that

$$c_0 = E(k, IV \oplus m_0) \quad c_i = E(k, c_{i-1} \oplus m_i) \qquad m_0 = D(k, m_0) \oplus IV \quad m_i = D(k, c_i) \oplus c_{i-1}$$

3. This exercise asks us to make proofs using games (you may want to check the tricks explained in the solution sheet of the first week).
Proving that f' is not secure reduces to searching a particular point x for which it is easy to correctly guess $b' = b$.
We should observe that in case the challenger picks the random F, the result we get is an element y completely random.

(a) $f'(k,x) = \begin{cases} f(k,x) & x \neq 0^n \\ k & \text{otherwise} \end{cases}$

Looking at the definition, for all $x \neq 0$, the output is the one of f and so secure.
So we can concentrate to the case $x = 0$.
The Adversary strategy is therefore to send $x = 0$ to the challenger

- If $b = 0$, k
- If $b = 1$, y

The key k is uniformly and randomly picked by the challenger and y is a random element. Querying 0, the attacker has $\frac{1}{2}$ the possibility that what he received is k.

Since he can't have any more clue, we conclude that f' is a secure PRF.

$$\text{Adv} = |\Pr(W_0) - \Pr(W_1)| = \left|\frac{1}{2} - \frac{1}{2}\right| = 0$$

(b) $f'((k_1,k_2),x) = f(k_1,x) \oplus f(k_2,x)$ Consider two different queries in the case $b = 0$:

$$f(k_1,x) \oplus f(k_2,x) \qquad f(k_1,y) \oplus f(k_2,y)$$

By assumption $f(k_i,x)$ appears as random elements, we are basically receiving outputs of f which is secure.
We can conclude that f' is a secure PRF.

$$\text{Adv} = |\Pr(W_0) - \Pr(W_1)| = \left|\frac{1}{2} - \frac{1}{2}\right| = 0$$

(c) $f'(k,x) = k \oplus x$
Consider two different queries in the case $b = 0$:

$$k \oplus x \qquad k \oplus y$$

If the second time we query $y = x \oplus k$, we get $k \oplus y = k \oplus (x \oplus k) = x$.
So an attacker can query the challenger with a random x. Let y be the value returned by $\mathcal{C}$. Then $\mathcal{A}$ can query $\mathcal{C}$ with y. If the answer is x, output $b' = 0$ else output $b' = 1$.
We can conclude that f' is not a secure PRF.

$$\text{Adv} = |\Pr(W_0) - \Pr(W_1)| = |1 - 0| = 1$$

Remark: even in the case $b = 1$, where we are querying a random function, we can get x as the result of our attack but this happens with really small (negligible, in fact) probability (with respect to the set of all the possible functions).

(d) $f'(k,x) = f(k, x \oplus 1^n)$ Consider two different queries in the case $b = 0$:

$$f(k, x \oplus 1^n) \qquad f(k, y \oplus 1^n)$$

At every query, x, the challenger replies with result from f on the *"inverse"* of our input (i.e., on x XORed with the all-ones string $\oplus 1^n$).
You should convince yourself that you are actually proving the security of f when the bits of the input are all flipped. We conclude that f' is a secure PRF.

$$\text{Adv} = |\Pr(W_0) - \Pr(W_1)| = \left|\frac{1}{2} - \frac{1}{2}\right| = 0$$

Formally, we should prove that *if* an adversary could break the security of f' , then $\mathcal{A}$ can be used to mount an attack against f. However, by hypothesis, f is secure, so no such $\mathcal{A}$ exists.
Imagine that we can prove that f' is not secure with an adversary $\mathcal{A}$.
Then we can use $\mathcal{A}$ against f:

- Start the security game with f
- Query for x
- The challenger output $y = f(k,x)$

- Ask A of the input $x \oplus 1$: this means that we are trying $f'(k, x \oplus 1) = f(k, x)$
- A will output us b' with an advantage
- We output this b'

We have so an advantage over f but this is impossible since we claimed that f is a secure PRF.

(e) $f'(k_2, x) = f(0^n, x) \| f(k_2, x)$ Consider two different queries in the case $b = 0$:

$$f(0^n, x) \| f(k_2, x) \qquad f(0^n, y) \| f(k_2, y)$$

Observe that we have a fixed part $f(0^n, \cdot)$. The attacker knows f (the only information missing to $\mathcal{A}$ is the key k_2) so he can just compute $f(0^n, x)$.
After the first query, he will compare the first part of the output and if it equals $f(0^n, x)$, he can output $b' = 0$.
We can conclude that f' is not a secure PRF.

$$\text{Adv} = |\Pr(W_0) - \Pr(W_1)| = |1 - 0| = 1$$

4. **Task:** hope you enjoyed creating a block cipher.

Medium

5. **Proof:** Let us consider two-block message $m_0 = (m, m), m_1 = (m, \hat{m})$ with $m \neq \hat{m}$.
We, the adversary, query the challenger on the two messages and gets $c = (c_1, c_2)$, which is an encryption of wither m_0 or m_1.
If $c_1 = c_2$, we are sure that the message was m_0 since the two encryptions are equal. If $c_1 \neq c_2$, then we can conclude that the message chosen by the challenger was m_0.
This proves that ECB is not semantic secure.

6. **Answers:**

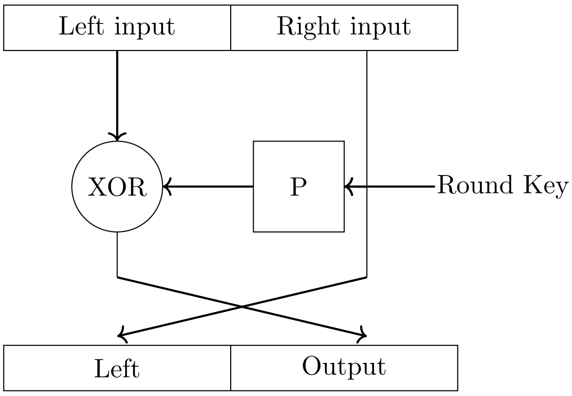
   - We have $n = 30$ and we compute a xor with 5 operations per second.
   Since the computational complexity is n^4, the total number of operation we have to do is $30^4 = 810000$. If we do 5 operation per second, we will need $810000/5 = 162000$ seconds which are 45 hours!
   - This time, we have that the algorithm complexity is n^2 and so the number of operations is $30^2 = 900$ if we perform 1 operation per second.
   We will need 900 seconds to complete our algorithm which is 15 minutes.

   Take away: this exercise should let you think about the difference between the *computational complexity* of an algorithm and the *computational power* of a calculator. If you can estimate the running time of an algorithm on your computer, you can decide if it makes sense to really run it.
   Just as a historical or side note reference, once upon a time, during my master thesis writing, I started an algorithm that I wrote and after 5 days I didn't get any result. I approximated the computation time and it was like ~ 10000 years.

7. **Definition:** A block cipher is a cipher (E, D) over $\mathcal{K}, \mathcal{M}, \mathcal{C}$ with $\mathcal{M} = \mathcal{C}$ for which

   - $D(k, E(k, m)) = m$ for all $m \in \mathcal{M}$
   - It has a *round* structure that is iterated for a fixed number r of time
   - It has a key-scheduler that expands the secret key k into r-round keys

   DES: DES is a 64 bit Feistel network with 16 rounds. The round of a Feistel network can be summarized with this structure:



DES is considered broken. **AES:** AES is a 11-round[1] substitution block cipher on 128 bit blocks.

AES's round is composed by four different algorithm:

- SubBytes
- ShiftRows
- MixColumns
- AddRoundKeys

AES is the actual encryption standard and it is considered secure.

The main differences between DES and AES are

- DES is a Feistel network and AES a substitution-cipher
- AES has double the block size
- DES is considered broken while AES not

You can also check out more details in lecture 03.

Hard

8. **Answer:** DES is computationally broken. This means that today's computational power can *"easily"* try every key for DES and find the exact one in a reasonable time.
   This does not mean that the algebraic construction is broken but just the fact that we can brute force every key.

9. **Proof:** We play the role of the adversary in the CPA game and use the following strategy:

   - The adversary asks for the encryption of one block of all zeros: query 1 is 0^n
   - The challenger encrypt the block and output $\hat{c} = (IV_1, \mathrm{Enc}(k, 0^n \oplus IV_1)$
   - Let $\hat{c} = (\hat{c}_1, \hat{c}_2)$
   - Suppose that the adversary has an algorithm to predict IV from IV_1
   - The adversary fixes $m_0 = IV \oplus IV_1$ and m_1 random message different from m_0. He sends m_0, m_1
   - The challenger flips a coin (i.e. choose a random bit as $\{0, 1\}$) and returns $c = (IV, \mathrm{Enc}(k, m_b \oplus IV) = (c_1, c_2)$
   - The adversary checks whether $c_2 = \hat{c_2}$ or not. If the equality holds, the adversary output $b' = 0$, otherwise $\mathcal{A}$ sets $b' = 1$

10. **Definition:** the Chosen Plaintext Attack security game is composed by two phases:

    (a) **Query phase**: the adversary asks the challenger to encrypt some messages m_i and gets the corresponding encryption c_i (usually $\mathcal{A}$ make polynomially (in the security parameter) many such queries)

[1] We generally always refer to the 128 bit AES.

(b) **Challenge phase**:

- The adversary choses m_0, m_1 such that neither m_0 nor m_1 have been queried in phase (a)
- The challenger flips a coin (i.e., $b \overset{R}{\leftarrow} \{0,1\}$) and encrypt m_b and return to $\mathcal{A}$ its ciphertext c_b
- The adversary output b' a guess for the b chosen by $\mathcal{C}$

11.

12. Let $c = DESX_{(k_1,k_2)}(m)$. We first xor both sides with k_2, which gives $c \oplus k_2 = DES_{k_1}(m \oplus k_2)$. Next, we apply DES decryption, to get $DES_{k_1}^{-1}(c \oplus k_2) = m \oplus k_2$. Finally, we again XOR both sides with k_2 to get the final result
$$m = DES_{k_1}^{-1}(c \oplus k_2) \oplus k_2.$$
This cipher can be attacked using a meet-in-the-middle attack.
We assume that the adversary has a few plaintext/ciphertext pairs (m, c). He can then do a brute force attack on the DES part, i.e. compute $x = DES_{k_1}(m)$ for all possible keys k_1, and store the resulting pairs (x, k_1) in a dictionary.
Then he goes through all possible k_2, computes $c \oplus k_2$ and looks it up in the dictionary. When found, he has a potential key pair (k_1, k_2).
The complexity of this attack is 2^{64}, which shows that the cipher does not provide 120 bits of security.

Think

12. No.
The reason is that there exists an algorithm which retrieves the secret key in 2^{126} operations while the brute force is 2^{128}. The attack strategy is of particular interest since it requires solely a random message m and its encryption c.
How does this prove that AES is not a perfect cipher?
The perfect secrecy equation essentially states that $\Pr(M = m|C = c) = \Pr(M = m)$.The previous equation means that *if you brute force the cipher, you **have** to check all the possible keys.*
However, the attack we refer to works without using all the possible keys. Essentially, you have a different probabilities on the left and on the right side of the equation, i.e., $\Pr(M = m|C = c) \neq \Pr(M = m)$. This proves that AES is not a perfect cipher.

Remark the consideration above does not imply that AES is broken. From a computational point of view, 2^{126} is still a unfeasible number of operations.

13. **Answer:** there are different reasons:
    - A block cipher should be secure. Taking a random permutation can be really risky because it can be, for example, linear and we want to achieve the maximal non-linearity.
    - Block cipher are implemented in hardware. Often cryptographers design ciphers that are a little *weaker* (from a security point of view) but which have optimal performance for applications
    - In Mathematics, every permutation in a finite set is a pseudo-random permutation.
    To explain this point, imagine that you randomly choose all the possible substitutions of your *random* function. When you have finished, you have a function that is deterministic because it can be represented in a clear mathematical way and so it is not random.

14. **Answer:** from a substitution function we can build (and maybe I miss something):
    - Pseudo-random generators (as you have seen in the exercises of the first week)
    - Block ciphers using
        - Feistel Network (last week construction)
        - Substitution-translation based (this week construction)
    - Stream ciphers (last week construction)
    - Block ciphers in ECB, CBC mode

- Pseudo-random functions using the block cipher construction
- A coin to flip

Quick summary: during the course you will encounter more applications/cryptographic functions that have a substitution function as a building block!
Observe that, in this exercise sheet, we did not ask you to prove that what you built is secure. Although you defined different cryptographic primitives, you are still far away from being able to prove their security!