Problems

3.1. As stated in Sect. 3.5.2, one important property which makes DES secure is that the S-boxes are nonlinear. In this problem we verify this property by computing the output of S_1 for several pairs of inputs.

Show that $S_1(x_1) \oplus S_1(x_2) \neq S_1(x_1 \oplus x_2)$, where "$\oplus$" denotes bitwise XOR, for:

1. $x_1 = 000000, x_2 = 000001$
2. $x_1 = 111111, x_2 = 100000$
3. $x_1 = 101010, x_2 = 010101$

3.2. We want to verify that $IP(\cdot)$ and $IP^{-1}(\cdot)$ are truly inverse operations. We consider a vector $x = (x_1, x_2, \ldots, x_{64})$ of 64 bit. Show that $IP^{-1}(IP(x)) = x$ for the first five bits of x, i.e. for x_i, $i = 1, 2, 3, 4, 5$.

3.3. What is the output of the first round of the DES algorithm when the plaintext and the key are both all zeros?

3.4. What is the output of the first round of the DES algorithm when the plaintext and the key are both all ones?

3.5. Remember that it is desirable for good block ciphers that a change in one input bit affects many output bits, a property that is called diffusion or the avalanche effect. We try now to get a feeling for the avalanche property of DES. We apply an input word that has a "1" at bit position 57 and all other bits as well as the key are zero. (Note that the input word has to run through the initial permutation.)

1. How many S-boxes get different inputs compared to the case when an all-zero plaintext is provided?
2. What is the minimum number of output bits of the S-boxes that will change according to the S-box design criteria?
3. What is the output after the first round?
4. How many output bit after the first round have actually changed compared to the case when the plaintext is all zero? (Observe that we only consider a single round here. There will be more and more output differences after every new round. Hence the term *avalanche effect*.)

3.6. An avalanche effect is also desirable for the key: A one-bit change in a key should result in a dramatically different ciphertext if the plaintext is unchanged.

1. Assume an encryption with a given key. Now assume the key bit at position 1 (prior to $PC-1$) is being flipped. Which S-boxes in which rounds are affected by the bit flip during DES encryption?
2. Which S-boxes in which DES rounds are affected by this bit flip during DES decryption?

3.7. A DES key K_w is called a *weak key* if encryption and decryption are identical operations:

$$\mathrm{DES}_{K_w}(x) = \mathrm{DES}^{-1}_{K_w}(x), \text{ for all } x \tag{3.1}$$

1. Describe the relationship of the subkeys in the encryption and decryption algorithm that is required so that Eq. (3.1) is fulfilled.
2. There are four weak DES keys. What are they?
3. What is the likelihood that a randomly selected key is weak?

3.8. DES has a somewhat surprising property related to bitwise complements of its inputs and outputs. We investigate the property in this problem.

We denote the bitwise complement of a number A (that is, all bits are flipped) by A'. Let $\oplus$ denote bitwise XOR. We want to show that if

$$y = \mathrm{DES}_k(x)$$

then

$$y' = \mathrm{DES}_{k'}(x'). \tag{3.2}$$

This states that if we complement the plaintext and the key, then the ciphertext output will also be the complement of the original ciphertext. Your task is to *prove* this property.

Try to prove this property using the following steps:

1. Show that for any bit strings A, B of equal length,

   $$A' \oplus B' = A \oplus B$$

   and

   $$A' \oplus B = (A \oplus B)'.$$

   (These two operations are needed for some of the following steps.)
2. Show that $PC-1(k') = (PC-1(k))'$.
3. Show that $LS_i(C'_{i-1}) = (LS_i(C_{i-1}))'$.
4. Using the two results from above, show that if k_i are the keys generated from k, then k'_i are the keys generated from k', where $i = 1, 2, \ldots, 16$.
5. Show that $IP(x') = (IP(x))'$.
6. Show that $E(R'_i) = (E(R_i))'$.
7. Using all previous results, show that if R_{i-1}, L_{i-1}, k_i generate R_i, then R'_{i-1}, L'_{i-1}, k'_i generate R'_i.
8. Show that Eq. (3.2) is true.

3.9. Assume we perform a known-plaintext attack against DES with one pair of plaintext and ciphertext. How many keys do we have to test in a worst-case scenario if we apply an exhaustive key search in a straightforward way? How many on average?

3.10. In this problem we want to study the clock frequency requirements for a hardware implementation of DES in real-world applications. The speed of a DES implementation is mainly determined by the time required to do one core iteration. This hardware kernel is then used 16 consecutive times in order to generate the encrypted output. (An alternative approach would be to build a hardware pipeline with 16 stages, resulting in 16-fold increased hardware costs.)

1. Let's assume that one core iteration can be performed in one clock cycle. Develop an expression for the required clock frequency for encrypting a stream of data with a data rate r [bit/sec]. Ignore the time needed for the initial and final permutation.
2. What clock frequency is required for encrypting a fast network link running at a speed of 1 Gb/sec? What is the clock frequency if we want to support a speed of 8 Gb/sec?

3.11. As the example of COPACOBANA [105] shows, key-search machines need not be prohibitive from a monetary point of view. We now consider a simple brute-force attack on DES which runs on COPACOBANA.

1. Compute the runtime of an average exhaustive key-search on DES assuming the following implementational details:
   - COPACOBANA platform with 20 FPGA modules
   - 6 FPGAs per FPGA module
   - 4 DES engines per FPGA
   - Each DES engine is fully pipelined and is capable of performing one encryption per clock cycle
   - 100 MHz clock frequency
2. How many COPACOBANA machines do we need in the case of an average search time of one hour?
3. Why does any design of a key-search machine constitute only an upper security threshold? By *upper security threshold* we mean a (complexity) measure which describes the maximum security that is provided by a given cryptographic algorithm.

3.12. We study a real-world case in this problem. A commercial file encryption program from the early 1990s used standard DES with 56 key bits. In those days, performing an exhaustive key search was considerably harder than nowadays, and thus the key length was sufficient for some applications. Unfortunately, the implementation of the key generation was flawed, which we are going to analyze. Assume that we can test 10^6 keys per second on a conventional PC.

The key is generated from a password consisting of 8 characters. The key is a simple concatenation of the 8 ASCII characters, yielding $64 = 8 \cdot 8$ key bits. With the permutation $PC-1$ in the key schedule, the least significant bit (LSB) of each 8-bit character is ignored, yielding 56 key bits.

1. What is the size of the key space if all 8 characters are randomly chosen 8-bit ASCII characters? How long does an average key search take with a single PC?
2. How many key bits are used, if the 8 characters are randomly chosen 7-bit ASCII characters (i.e., the most significant bit is always zero)? How long does an average key search take with a single PC?
3. How large is the key space if, in addition to the restriction in Part 2, only letters are used as characters. Furthermore, unfortunately, all letters are converted

to capital letters before generating the key in the software. How long does an average key search take with a single PC?

3.13. This problem deals with the lightweight cipher PRESENT.

1. Calculate the state of PRESENT-80 after the execution of one round. You can use the following table to solve this problem with paper and pencil. Use the following values (in hexadecimal notation):
   plaintext = `0000 0000 0000 0000`,
   key = `BBBB 5555 5555 EEEE FFFF`.

Plaintext	`0000 0000 0000 0000`
Round key	
State after KeyAdd	
State after S-Layer	
State after P-Layer	

2. Now calculate the round key for the second round using the following table.

Key	`BBBB 5555 5555 EEEE FFFF`
Key state after rotation	
Key state after S-box	
Key state after CounterAdd	
Round key for Round 2	