Cryptography: Problem Set #1

Due on Monday, June 2nd at 10:00pm

Elena Machkasova

Ellis Weglewski

Problem 1

(3.1) Question:
Show that $S_1(x_1) \oplus S_1(x_2) \neq S_1(x_1 \oplus x_2)$ for:

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

Solutions:

1. (a) $(S_1((x_1 = 0, 0 = 14) = 1110 \oplus S_1((x_2 = 0, 1 = 00 = 0000)) = 1110$
   (b) $S_1(000000 \oplus 0000001) = S_1(000001) = 0000$
   (c) $0000 \neq 1110$
2. (a) $(S_1(x_1 = 15, 3 = 13) = 1101 \oplus S_1(x_2 = 0, 2 = 04) = 0100) = 1001$
   (b) $S_1(111111 \oplus 100000) = S_1(011111) = 1000$
   (c) $1001 \neq 1000$
3. (a) $(S_1(x_1 = 5, 2 = 06) = 0110 \oplus S_1(x_2 = 10, 1 = 12) = 1100) = 1010$
   (b) $S_1(101010 \oplus 010101) = S_1(111111) = 1101$
   (c) $1010 \neq 1101$

Problem 2

(3.2) Question:
We want to verify that $IP(\cdot)$ and $IP^{-1}(\cdot)$ are truly inverse operations. We consider a vector $x = (x_1, x_2, ..., x_{64})$ of 64 bit. Show that $IP^{-1}(IP(x)) = x$ for the first five bits of x,i.e. for $x_i = 1, 2, 3, 4, 5$.
Solution:
Via pg. 70: $IP(Y) = IP(IP^{-1}(R_{16}L_{16}))$. I take this to imply: $IP^{-1}(Y) = IP^{-1}(IP(R_{16}L_{16}))$. If we look at IP and IP^{-1} boxes on pg 62, we can see where each byte is sent after each operation. The byte in position 1 is sent to position 40 after IP and the byte in position 40 is sent to position 1 after IP^{-1}. This means that IP^{-1} is undoing what IP did which makes them mutually inverse. We can construct a flow chart to illustrate where each byte goes in each step of the operation $IP^{-1}(IP(x)) = x$:

1. $x_1 \rightarrow x_{40} \rightarrow x_1$
2. $x_2 \rightarrow x_8 \rightarrow x_2$
3. $x_3 \rightarrow x_{48} \rightarrow x_3$
4. $x_4 \rightarrow x_{16} \rightarrow x_4$
5. $x_5 \rightarrow x_{56} \rightarrow x_5$

Problem 3

(3.3) Question:
What is the output of the first round of the DES algorithm when the plaintext and the key are both all zeros?
Solution:
If the plaintext is all zeros than all the permutations and expansions will be composed of entirely zeros as well. As far as the key and its subkeys go, they will all be zeros too. This leaves us only needing to plug zero (000000) into each S-Box, permute the result, and then XOR with the 32 zeros on the left:

1. $S_1(000000) = 14 = 1110$
2. $S_2(000000) = 15 = 1111$
3. $S_3(000000) = 10 = 1010$
4. $S_4(000000) = 07 = 0111$
5. $S_5(000000) = 02 = 0010$
6. $S_6(000000) = 12 = 1100$
7. $S_7(000000) = 04 = 0100$
8. $S_8(000000) = 13 = 1101$

The output of the S-Boxes is: 11101111101001110010110001001101 which still needs to be permuted and XOR'd. Through consulting the permutation table on page 66, our new right side is 11011000110110001101101110111100. XOR with all zeros would give us the same thing so that is our answer.

Problem 4

(3.7) Question:
A DES key K_w is called a weak key if encryption and decryption are identical operations:

$$DES_{k_w}(x) = DES^{-1}_{k_w}(x),\text{for all } x$$

1. Describe the relationship of the subkeys in the ecnryption and decryption algorithm that is required so that above-defined equality is fulfilled.
2. There are four weak DES keys. What are they?
3. What is the likelihood that a randomly selected key is weak?

Solution:

1. In order to satisfy the above statement about weak keys, the sub-keys in each step of the encryption operation would have to be the same as the sub-keys in each step of the decryption operation. Given that the decryption operation is the same as the encryption operation, just with a reversed key schedule, and the fact that $C_0 = C_{16}$ and $D_0 = D_{16}$, the required relationship can be defined as:

   $$k_{i+1} = k_{16-i}\text{for } i = 0, 1, ...7$$

   This works because at 7 the keys will meet in the middle meaning they are all the same.

Problem 4 continued on next page. . .

2. The key schedule essentially splits the initial key value in to 2 halves and permutes them independently. In order to satisfy the above statement about weak keys, each half would have to be composed of entirely either ones or zeros so that the key is the same after each permutation. Thus:
   (a) 00000000000000
   (b) 00000001111111
   (c) 11111110000000
   (d) 11111111111111
3. The keys are 56 bits which means that there are 2^{56} possible keys. The chances of picking one of these 4 weak keys would be $4/2^{56}$.

Problem 5

(3.9) Question:
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?
Solution:
Given that there are 2^{56} possible keys in DES, one would have to test them all in the worst case. So the worst case is 2^{56}. I googled "average case of an exhaustive key attack" and found that the average case is approximately half of the worst case time. So the average amount of keys one would have to test is $2^{56}/2 = 2^{55}$.

Problem 6

(3.12) Question:
We study a real-world case in this problem. A commercial file ecryption 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 goin 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 $64 = 8 * 8$ 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. Futhermore, 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?

Solution:

1. We can build off our answer to the previous question to answer this. The key space would be 2^{56} bits. The average amount of keys check in a key search is 2^{55} and given that we can test 10^6 keys per second, the average key search would take $2^{55}/10^6$ seconds.

2. If the key its composed of 8 randomly chosen 7-bit ASCII characters then the least significant bit would be the seventh. We would drop this bit which would give us a key bit size of 48 bits. We can then go through the same steps as the previous part and discern that there are 2^{48} possible keys and the average key search would take $2^{47}/10^6$ seconds.

3. Given that there are only 26 letters, this would mean that there are only $26 * 8 = 208$ possible keys. The average key search would test 104 keys and it would take $104/10^6$ seconds.