Problem No. 1

Let e_K be the encryption transformation of a block cipher with 64-bit key K and a 64-bit block length. The key size of the block cipher is doubled as follows. Given two 64-bit keys K_1 and K_2 and 64-bit plaintext “x” the ciphertext “y” is computed as $y=e_{K2}(x \oplus K_1)$. Assume that an attacker has two known plaintext-ciphertext pairs (x_1, y_1) and (x_2, y_2) encrypted in this manner with 128-bit key (K_1, K_2). Show that how an attacker can find the used 128-bit key with a large probability by doing the exhaustive search over a 64-bit partial key.

Solution to Problem No. 1

If we decrypt y_1 and y_2 with the correct partial key K_2, we get $d_{K2}(y1) = x_1 \oplus K_1$ and $d_{K2}(y2) = x_2 \oplus K_1$. Hence for the correct key K_2 we have $d_{K2}(y_1) \oplus d_{K2}(y_2) = x_1 \oplus x_2$: Based on this observation we can test for a 64-bit key candidate Z whether the equality $d_Z(y_1) \oplus d_Z(y_2) = x_1 \oplus x_2$: holds.

The correct key $Z = K_2$ always passes the test. This solution is unique if none of the other candidates passes the test. When estimating the probability that the solution is unique, we may assume that for a wrong key Z the value $d_Z(y_1) \oplus d_Z(y_2)$ is a value that is selected uniformly at random.

The probability that it equals $x_1 \oplus x_2$ can therefore be assumed to be 2^{-64}. Hence the probability that none of the 2^{64} -1 wrong keys hits $x_1 \oplus x_2$ is approximately equal to $(1\text{-}1/2^{64})^{264\text{-}1} \approx e^{-1}$: After K_2 is found, then K_1 can be computed as $K_1 = dK_2(y_1) \oplus x_1$.

Problem No. 2

Let “K” be a 56-bit DES key, let “L” be a 64-bit string, and let “M” be a 64-bit plaintext.

Let $\text{DESY}(K \| L, M) = \text{DES}(K, L \oplus M)$

$\text{DESW}(K \| L, M) = L \oplus \text{DES}(K, M)$:

This defines block ciphers DESY, DESW: $\{0, 1\}^{120} \times \{0, 1\}^{64} \rightarrow \{0, 1\}^{64}$.

Present the best possible key-recovery attacks that you can of these block ciphers. Your attacks should use very few input-output examples, not more than three. State the running time of your attacks.

Solution to Problem No. 2

Note that $C = \mathsf{DESY}(K \| L, M)$ iff $\mathrm{DES}^{-1}(K, C) \oplus M = L$. This leads to the following key-recovery attack:

Adversary $A_{\mathsf{DESY}}((M_1, C_1), (M_2, C_2), (M_3, C_3))$
    for each key $T \in \{0,1\}^{56}$ **do**
        $L_1 \leftarrow \mathrm{DES}^{-1}(T, C_1) \oplus M_1$; $L_2 \leftarrow \mathrm{DES}^{-1}(T, C_2) \oplus M_2$; $L_3 \leftarrow \mathrm{DES}^{-1}(T, C_3) \oplus M_3$
        if $L_1 = L_2 = L_3$ then **return** $T \| L_1$

The time taken by this attack is that of about $3 \cdot 2^{56}$ DES^{-1} computations.

Note that $C = \mathsf{DESW}(K \| L, M)$ iff $C \oplus \mathrm{DES}(K, M) = L$. This leads to the following key-recovery attack:

Adversary $A_{\mathsf{DESW}}((M_1, C_1), (M_2, C_2), (M_3, C_3))$
    for each key $T \in \{0,1\}^{56}$ **do**
        $L_1 \leftarrow \mathrm{DES}(T, M_1) \oplus C_1$; $L_2 \leftarrow \mathrm{DES}(T, M_2) \oplus C_2$; $L_3 \leftarrow \mathrm{DES}(T, M_3) \oplus C_3$
        if $L_1 = L_2 = L_3$ then **return** $T \| L_1$

The time taken by this attack is that of about $3 \cdot 2^{56}$ DES computations.

Problem No. 3

- Consider a Feistel-type block cipher algorithm with the properties:
    - Input, output block length is 32 bits and key size is 32 bits.
    - Given a key K, the key scheduling requires 10 microseconds.
    - After the key scheduling produces all sub-keys, the encryption or decryption of a single block requires 10 nanoseconds.
- Compute the total time required to encrypt 1kilobits of data.

Solution to Problem No. 3

Compute the total time required to encrypt 1 kilobits of data.

1 kbit = 1024 bits, Divide 1024 bits into blocks with 32 bits size (1024/32)

Encrypt each of the 32 blocks of data .

$= 3.2 \times 10^{-7}$ seconds.

$1 \times 10^{-5} + 3.2 \times 10^{-7} = 1 \times 10^{-5} + 0.032 \times 10^{-5} = 1.032 \times 10^{-5}$ seconds

Problem No. 4

A simple synchronous stream cipher with an eight bit key works as follows:

The key is interpreted as two four-bit natural numbers k_0 and k_1. We form a state machine with a four-bit state initialized to k_0 that produces the key-stream as follows:

1. Output the state to the key-stream as four bits, most significant bit first.
2. Update the state by adding k_1 modulo 16.
3. Go to 1.

Encryption is done by xor-ing successive plaintext bits with bits from the key-stream. Encrypt the word **BANANA** with the key **57hex**. Assume ASCII coding of plain text.

Solution to Problem No. 4

The message is six characters, which is 48 bits when we use (8-bit) ASCII. We therefore need 12 four bit chunks of output from the keystream (we use hex notation):

- o0 = k0 = 5
- o1 = o0 + k1 = 5 + 7 = c
- o2 = o1 + k1 = c + 7 = 3
- o3 = o2 + k1 = 3 + 7 = a
- o4 = o3 + k1 = a + 7 = 1
- o5 = o4 + k1 = 1 + 7 = 8
- o6 = o5 + k1 = 8 + 7 = f
- o7 = o6 + k1 = f + 7 = 6
- o8 = o7 + k1 = 6 + 7 = d
- o9 = o8 + k1 = d + 7 = 4
- o10 = o9 + k1 = 4 + 7 = b
- o11 = o10 + k1 = b + 7 = 2

Thus the plaintext and the keystream are

0100 0010 0100 0001 0100 1110 0100 0001 0100 1110 0100 0001

0101 1100 0011 1010 0001 1000 1111 0110 1101 0100 1011 0010

Finally, bitwise xor gives the ciphertext

0001 1110 0111 1011 0101 0110 1011 0111 1001 1010 1111 0011

Problem No. 5

- For convenience, you set up your email program to automatically decrypt your incoming encrypted mail, and to quote the plain text in your replies. To what kind of cryptographic attack might you be opening yourself? Justify your answer.

Solution to Problem No. 5

A chosen-ciphertext attack.