Prof. Dr. Rudolf Mathar, Dr. Arash Behboodi, Jose Leon

Exercise 6
- Proposed Solution -

Friday, June 3, 2016

Solution of Problem 1

a) The bit error occurs in block C_i, $i > 0$, with block size BS.

mode	M_i	max #err	remark
ECB	$E_K^{-1}(C_i)$	BS	only block C_i is affected
CBC	$E_K^{-1}(C_i) \oplus C_{i-1}$	BS+1	C_i and one bit in C_{i+1}
OFB	$C_i \oplus Z_i$	1	one bit in C_i, as $Z_0 = C_0, Z_i = E_K(Z_{i-1})$
CFB	$C_i \oplus E_k(C_{i-1})$	BS+1	C_i and one bit in C_{i+1}
CTR	$C_i \oplus E_K(Z_i)$	1	one bit in C_i, $Z_0 = C_0, Z_i = Z_{i-1} + 1$

b) If one bit of the ciphertext is lost or an additional one is inserted in block C_i at position j, all bits beginning with the following positions may be corrupt:

mode	block	position
ECB	i	1
CBC	i	1
OFB	i	j
CFB	i	j
CTR	i	j

In ECB and CBC, all bits of blocks C_i, C_{i+1} may be corrupt.
In OFB, CFB, CTR, all bits beginning at position j of block C_i may be corrupt.

Solution of Problem 2

$$\begin{pmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \end{pmatrix} = \begin{pmatrix} x & (x+1) & 1 & 1 \\ 1 & x & (x+1) & 1 \\ 1 & 1 & x & (x+1) \\ (x+1) & 1 & 1 & x \end{pmatrix} \cdot \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} \in \mathbb{F}_{2^8}^4 \tag{1}$$

It is to show that:

$$(c_3u^3 + c_2u^2 + c_1u + c_0)((x+1)u^3 + u^2 + u + x) \equiv \sum_{i=0}^{3} r_i u^i \pmod{(u^4+1)}. \tag{2}$$

We expand the multiplication on the left hand side of (2), reduce it modulo $u^4 + 1 \in \mathbb{F}_{2^8}[u]$, and use the abbreviations $(r_0, r_1, r_2, r_3)'$ according to (1).

$$
\begin{aligned}
& (c_3u^3 + c_2u^2 + c_1u + c_0)((x+1)u^3 + u^2 + u + x) \\
= & c_3(x+1)u^6 + c_3u^5 + c_3u^4 + c_3xu^3 + \\
& c_2(x+1)u^5 + c_2u^4 + c_2u^3 + c_2xu^2 + \\
& c_1(x+1)u^4 + c_1u^3 + c_1u^2 + c_1xu + \\
& c_0(x+1)u^3 + c_0u^2 + c_0u + c_0x \\
= & [c_3(x+1)]u^6 + [c_3 + c_2(x+1)]u^5 + [c_3 + c_2 + c_1(x+1)]u^4 \\
& + [c_3x + c_2 + c_1 + c_0(x+1)]u^3 + [c_2x + c_1 + c_0]u^2 + [c_1x + c_0]u + c_0x.
\end{aligned}
$$

Now, we apply the modulo operation and merge terms:

$$
\begin{aligned}
\equiv & [c_3x + c_2 + c_1 + (x+1)c_0]u^3 + [c_3(x+1) + c_2x + c_1 + c_0]u^2 + \\
& [c_3 + c_2(x+1) + c_1x + c_0]u + [c_3 + c_2 + c_1(x+1) + c_0x] \\
\overset{(1)}{\equiv} & r_3u^3 + r_2u^2 + r_1u + r_0 \equiv \sum_{i=0}^{3} r_iu^i \pmod{(u^4+1)}
\end{aligned}
$$

Solution of Problem 3

The given AES-128 key is denoted in hexadecimal representation:

$$K = (2D\ 61\ 72\ 69 \mid 65\ 00\ 76\ 61 \mid 6E\ 00\ 43\ 6C \mid 65\ 65\ 66\ 66)$$

(a) The round key is $K_0 = K = (W_0\ W_1\ W_2\ W_3)$ with $W_0 = (2D\ 61\ 72\ 69)$, $W_1 = (65\ 00\ 76\ 61)$, $W_2 = (6E\ 00\ 43\ 6C)$, $W_3 = (65\ 65\ 66\ 66)$.

(b) To calculate the first 4 bytes of round key K_1 recall that $K_1 = (W_4\ W_5\ W_6\ W_7)$. Follow Alg. 1 as given in the lecture notes to calculate W_4:

	W_0	2	D	6	1	7	2	6	9
$\oplus$	tmp	4	C	3	3	3	3	4	D
	W_0	0010	1101	0110	0001	0111	0010	0110	1001
$\oplus$	tmp	0100	1100	0011	0011	0011	0011	0100	1101
	W_4	0110	0001	0101	0010	0100	0001	0010	0100
	W_4	6	1	5	2	4	1	2	4

Algorithm 1 AES key expansion (applied)

for $i \leftarrow 4;\ i < 4 \cdot (r+1);\ i++$ **do**
  Initialize *for*-loop with $i \leftarrow 4$. We have $r = 1$ for K_1.
  tmp $\leftarrow W_{i-1}$
  tmp $\leftarrow W_3 = (65\ 65\ 66\ 66)$
  if $(i \mod 4 = 0)$ **then**
    result is *true* as $i = 4$.
    tmp $\leftarrow$ `SubBytes`(`RotByte`(tmp)) $\oplus$ `Rcon`$(i/4)$
    Evaluate this operation step by step:
    `RotByte`(tmp) $= (65\ 66\ 66\ 65)$, i.e., a cyclic left shift of one byte
    To compute `SubBytes`(65 66 66 65) evaluate Table 5.8 for each byte:
    $(\text{row } 6, \text{col } 5)$ provides $77_{10} = 4D_{16}$
    $(\text{row } 6, \text{col } 6)$ provides $51_{10} = 33_{16}$
    Note that the indexation of rows and columns starts with zero.
    `SubBytes`$(65\ 66\ 66\ 65) = (4D\ 33\ 33\ 4D)$
    $i/4 = 1$
    `Rcon`$(1) = ($`RC`$(1)\ 00\ 00\ 00)$, with `RC`$(1) = x^{1-1} = x^0 = 1 \in \mathbb{F}_{2^8}$.
    tmp $\leftarrow (4D\ 33\ 33\ 4D) \oplus (01\ 00\ 00\ 00) = (4C\ 33\ 33\ 4D)$
  end if
  $W_i \leftarrow W_{i-4} \oplus \text{tmp}\ W_4 \leftarrow W_0 \oplus \text{tmp}$. Then, next iteration, $i \leftarrow 5$...
end for

Solution of Problem 4

The following procedure relies on a brute-force attack to obtain the keys K_1 and K_2:

1. Fix m and compute $c = E_{K_1}(E_{K_2}(E_{K_2}(m)))$, i.e., perform a chosen-plaintext attack.
2. Generate a list of encrypted ciphertexts $E_k(E_k(m))$ for the fixed m, where k runs through all possible keys.
3. Generate another list of deciphered plaintexts $D_{k'}(c)$ for the fixed c, where k' runs through all possible keys.
4. A match between the two lists is a pair of keys (k, k') with $E_{k'}(E_k(E_k(m))) = c$. There should only be a small number of such pairs.

For each pair (k, k'), choose another plaintext m' and check if it produces the corresponding ciphertext c'. This should eliminate most of the incorrect pairs. Repeating this procedure a few times should yield the correct pair $(k, k') = (K_1, K_2)$ with increasing probability.