Modes of Operation (Reprise)

Block ciphers can be applied to longer ciphertexts using one of various modes of operation. We assume that the input is plaintext $M = M_1M_2 \ldots$, the block enciphering map for given key $K$ is $E_K$, and the output is $C = C_1C_2 \ldots$. The following gives a summary of the major modes of operation.

**Electronic Codebook Mode.** For a fixed key $K$, the output ciphertext is given by $C_j = E_K(M_j)$ with output $C_1C_2 \ldots$.

**Ciphertext Block Chaining Mode.** For input key $K$, and initialization vector $C_0$, the output ciphertext is given by $C_j = E_K(C_{j-1} \oplus M_j)$, with output $C_0C_1C_2 \ldots$.

**Ciphertext Feedback Mode.** Given plaintext $M_1M_2 \ldots$ in $r$-bit blocks, a key $K$, an $n$-bit cipher $E_K$, and an $n$-bit initialization vector $I = I_1$, the ciphertext is computed as:

$$ C_j = M_j \oplus L_r(E_K(I_j)) $$

$$ I_{j+1} = R_{n-r}(I_j) \| C_j $$

where $R_{n-r}$ and $L_r$ are the operators which take the right-most $n-r$ bits and the left-most $r$ bits, respectively, and $\|$ is concatenation.

**Output Feedback Mode.** Given plaintext $M_1M_2 \ldots$ in $r$-bit blocks, a key $K$, an $n$-bit cipher $E_K$, and an $n$-bit initialization vector $I = I_0$, the ciphertext is computed as:

$$ I_j = E_K(I_{j-1}) $$

$$ C_j = M_j \oplus L_r(I_j), $$

where $L_r$ is the operator which takes the left-most $r$ bits.

1. What mode of operation has been used in the assignment and in class up to this point, and why? What are the security disadvantages of this mode of operation?
   
   **Solution** Electronic codebook mode – it leaves the cipher most open to analysis of its statistical properties, so that we can demonstrate the methods to crack it. It is also the most natural and naive way to apply a block cipher.

2. Let $E_K$ be the 4-bit cipher defined by:

$$ E_K(M) = (K \oplus M) \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} = (X_1 + X_3, X_2 + X_4, X_2 + X_3, X_1 + X_4) $$
where \( X = K \oplus M = (X_1, X_2, X_3, X_4) \). Encipher the message \( M \) given by

\[
11010110110011001000100100010000,
\]

using the key \( K = 1011 \), in (i) ECB mode, in (ii) CBC mode with initialization vector 1001, and in (iii) CFB mode with initialization vector 1001 and \( r = 1 \).

**Solution** The ciphertext output for each of ECB, CBC, and CFB modes is:

- **ECB mode:** \( 11001010001111111100000000001111 \)
- **CBC mode:** \( 0000101000001111111100011100001111 \)
- **CFB mode:** \( 0000011110100001111111100010110001 \)

Here the leading initialization vector 1001 is omitted in the CBC output. These calculations are achieved with the code on the subsequent page.

3. How many steps are required for error recovery from a ciphertext transmission error in ECB and CBC modes?

**Solution** The blocks in ECB mode are independent, so error recovery is immediate, i.e. an error affects only the block in which it occurs. In CBC mode recovery from errors occurs after two blocks.

4. If \( n = 64 \) and \( r = 8 \), how many steps in CFB mode does it take to recover from an error in a ciphertext block? What about in OFB mode?

**Solution** Recovery in CFB mode occurs after \( \lceil n/r \rceil = 64/8 = 8 \) blocks. In OFB mode recovery is immediate, provided synchronization is not lost.

```magma
// // --Appendix of Magma Code--
//
// Define shorthand notation:
ZZ := Integers();
I2S := IntegerToString;
function EK(K,M)
    X := [ M[i]+K[i] : i in [1..4] ];
end function;

Kbits := [GF(2)|1,0,1,1];
Mbits := [GF(2) |
    1,1,0,1,0,1,1,0,
    1,1,0,0,1,1,1,0,
    0,0,1,0,0,1,0,0,
    0,1,0,0,1,0,0,0 ];
```
// ECB Mode Computation
ECBbits := &cat[EK(Kbits,Mbits[[4*i+1..4*i+4]]) : i in [0..7]]; "ECB mode:" &cat[ I2S(ZZ!x) : x in ECBbits ];

// CBC Mode Computation
Cbits := [GF(2)|1,0,0,1];
CBCbits := [GF(2)|];
for i in [0..7] do
    CBCbits cat:= Cbits;
end for;
"CBC mode:" &cat[ I2S(ZZ!x) : x in CBCbits ];

// CFB Mode Computation
Ibits := [GF(2)|1,0,0,1];
CFBbits := [GF(2)|];
for j in [1..32] do
    Append(~CFBbits,Mbits[j]+EK(Kbits,Ibits)[1]);
    Ibits := Ibits[[2..4]] cat [CFBbits[j]];
end for;
"CFB mode:" &cat[ I2S(ZZ!x) : x in CFBbits ];