# TTM4135 Worksheet 3: Block ciphers and modes of operation

Tjerand Silde and Emil August Hovd Olaisen
{tjerand.silde, emil.august.olaisen}@ntnu.no

Spring 2026

1. Review the definitions of the following concepts:
   (a) confusion and diffusion;
   (b) product cipher and iterated cipher;
   (c) Feistel cipher;
   (d) substitution-permutation network;
   (e) ECB mode;
   (f) CBC mode;
   (g) CTR mode;
   (h) true random number generator (TRNG) and pseudorandom number generator (PRNG).

2. Suppose a certain system can perform $2^{55}$ trial encryptions of a DES block in 1 second.
   (a) Assuming that computational power remains the same, how may years should it take, on average, for this system to perform a brute force attack on two-key triple DES?
   (b) Moore's law says that computers double their computational power every two years. Assuming this continues for ever, how long will it be before two-key triple DES can be broken within one day?

**Solution:**

(a) There are $2^{112}$ keys to eliminate so the cryptanalyst needs to search through $2^{111}$ of these on average. But each key trial now requires two encryptions and one decryption, so that means the time of $3 \times 2^{111}$ encryptions on average (since encryption and decryption take about the same time). Therefore the attack should take around $3 \times 2^{111}/2^{55}$ seconds = $3 \times 2^{56}$ seconds. There are 31536000 seconds in a (non-leap) year which is around $2^{25}$ seconds. Therefore the search takes around $3 \times 2^{31}$ years, or around 6 billion years.

(b) Moore's law allows to compute twice as many encryptions in the same time after two years. There are roughly $2^{16}$ seconds in a day so we can perform $2^{55} \times 2^{16} = 2^{55+16} = 2^{71}$ encryptions in one day at present (according to the assumption). It will be $2 \times (111 - 71) = 80$ years before we can compute $2^{111}$ computations in a day. Two or three years more would be required before we are able to compute $3 \times 2^{111}$ encryptions within one day.

3. Consider a block cipher with encryption function $E$ and decryption function $D$. Suppose that this cipher is *linear* with respect to messages:
$$E(M_1, K) \oplus E(M_2, K) = E(M_1 \oplus M_2, K)$$
for any messages $M_1$ and $M_2$ and any fixed key $K$. Suppose that the block size is 128 bits. Show that an attacker can use a chosen ciphertext attack, with just 128 chosen ciphertexts, to easily decrypt any message.

**Solution:**

First we show that decryption is also linear. Let $C_1$ and $C_2$ be any two ciphertexts. Then there exist two plaintexts $M_1$ and $M_2$ with $C_1 = E(M_1, K)$ and $C_2 = E(M_2, K)$ and we get:

$$\begin{aligned} D(C_1 \oplus C_2, K) &= D(E(M_1, K) \oplus E(M_2, K), K) \\ &= D(E(M_1 \oplus M_2, K), K)) \\ &= M_1 \oplus M_2 \\ &= D(C_1, K) \oplus D(C_2, K) \end{aligned}$$

Furthermore, this can be extended to any number of ciphertexts:

$$D(C_1 \oplus C_2 \oplus \ldots \oplus C_n, K) = D(C_1, K) \oplus D(C_2, K) \oplus \ldots \oplus D(C_n, K)$$

Now suppose that the attacker obtains the decryptions of the following 128 ciphertexts:

$$\begin{aligned} C_0 &= (1, 0, 0, \ldots, 0) \\ C_1 &= (0, 1, 0, \ldots, 0) \\ &\vdots \\ C_{127} &= (0, 0, 0, \ldots, 1) \end{aligned}$$

Given any ciphertext $C^*$, the attacker can write it as

$$C^* = a_0 C_0 \oplus a_1 C_1 \oplus a_2 C_2 \oplus \ldots \oplus a_{127} C_{127}$$

where the $a_i$ are the bits of the ciphertext, and so by the linearity of decryption

$$D(C^*, K) = a_0 D(C_0, K) \oplus a_1 D(C_1, K) \oplus a_2 D(C_2, K) \oplus \ldots \oplus a_{127} D(C_{127}, K).$$

Since each of these decryptions is assumed to be known, the attacker can just add together the plaintexts corresponding to the bits of $C^*$, that is the $D(C_i, K)$ for which $a_i = 1$.

4. Consider the following two (toy) ciphers. Both ciphers are two-round iterated block ciphers with a block length of 8 bits and a key length of 8 bits. In each case work through the steps required to encrypt the plaintext block $P = 01010101$ using key $K = 11000011$.

   (a) A substitution-permutation in which each S-box operates on sub-blocks of 2 bits. Thus $m = 4$ and $l = 2$. The key schedule is defined by $K_1 = K$ and $K_2 = K_1 << 1$ where $<< 1$ denotes a cyclic shift left by one position. The cipher conduct the following operations in order:

      1. XOR the block with the appropriate subkey.
      2. Apply the permutation $\pi_S$, which is defined by the following table:

| Input block | 00 | 01 | 10 | 11 |
|---|---|---|---|---|
| Output block | 10 | 00 | 01 | 11 |

      3. Apply the permutation $\pi_P$, which is defined by the following table, where the block bits are labeled from 0 to 7:

| Input position | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| Output position | 3 | 6 | 0 | 5 | 2 | 1 | 7 | 4 |

   (b) A Feistel cipher with:

- $f$ function defined by $f(X, K_i) = X \vee K_i$ for any half block $X$ and subkey $K_i$. Here the $\vee$ operation is done on each bit separately so that if $K_i = (k_0, k_1, k_2, k_3)$ and $X = (x_0, x_1, x_2, x_3)$ then $f(X, K_i) = (k_0 \vee x_0), (k_1 \vee x_1), (k_2 \vee x_2), (k_3 \vee x_3)$.
- key schedule defined by $K_1$ is the 4 leftmost bits of $K$ and $K_2$ is the 4 rightmost bits of $K$.

**Solution:**

(a) The subkeys are $K_1 = 11000011$ and $K_2 = 10000111$.

**Round 1**

Step 1: XOR in the subkey: $01010101 \oplus 11000011 = 10010110$

Step 2: Perform the S-box substitutions: $10 \rightarrow 01$, $01 \rightarrow 00$, $01 \rightarrow 00$, $10 \rightarrow 01$. So the new state is: 01000001.

Step 3: Perform the bit position permutation: the bit in position 0 (0) becomes the new bit in position 3 and so on. The new state is 00001010.

**Round 2**

Step 1: XOR in the subkey: $00001010 \oplus 10000111 = 10001101$

Step 2: Perform the S-box substitutions: $10 \rightarrow 01$, $00 \rightarrow 10$, $11 \rightarrow 11$, $01 \rightarrow 00$. So the new state is: 01101100.

Step 3: Perform the bit position permutation. The final state is 11100010.

(b) The subkeys are $K_1 = 1100$ and $K_2 = 0011$. We set $L_0 = 0101$ and $R_0 = 0101$.

**Round 1** $L_1 = R_0 = 0101$

$R_1 = L_0 \oplus f(R_0, K_1) = 0101 \oplus [R_0 \vee K_1] = 0101 \oplus (0101 \vee 1100) = 0101 \oplus 1101 = 1000$

**Round 2** $L_2 = R_1 = 1000$

$R_2 = L_1 \oplus f(R_1, K_2) = 0101 \oplus [R_1 \vee K_2] = 0101 \oplus [1000 \vee 0011] = 0101 \oplus 1011 = 1110$

The final output is 10001110.

5. Consider the use of double encryption applied to the AES algorithm with two 128-bit keys. How much storage and computation would be required to execute a *meet-in-the-middle* attack (as described for DES in the lecture)?

**Solution:** We assume that the attacker has a single plaintext-ciphertext block pair, $(P, C)$ so that $E(E(P, K_1), K_2) = C$ where $E$ denotes single AES encryption and $K_1, K_2$ are the two keys used in double encryption. The attacker:

(a) computes $E(P, K)$ for all keys $K$ and stores the results.

(b) computes $D(C, K')$ for keys $K'$ until a match is found in the stored results.

The attack requires $2^{128}$ AES encryption and storage of the resulting $2^{128}$ ciphertext in part (a). Up to $2^{128}$ AES decryptions are required for part (b).

In order to make the comparison easier it is probably necessary to order the outputs from part (a). This will increase the computation required in practice, but allows for a fast binary search in part (b).

Note that this amount of computation is currently quite infeasible. However, it is far easier than the brute force attack of searching through $2^{256}$ keys used in double AES encryption.

6. Suppose we want to encrypt more than one block of random bits using ECB mode with a block cipher, but without padding. This can be achieved with a technique known as *ciphertext stealing*. For example, suppose we encrypt a 200-bit random key using AES in ECB mode with key $K$. The plaintext is two blocks $M_1, M_2$ where $M_2$ is a 72-bit 'short' block. Then we compute:

$$\begin{aligned} C_2 \parallel J &= E(M_1, K) \\ C_1 &= E(M_2 \parallel J, K) \end{aligned}$$

and send $(C_1, C_2)$ to the receiver. Here $J$ are the 56 last bits of the encryption of $M_1$ under $K$.

(a) What is the length of the ciphertext?

(b) Show how the message can be decrypted back to the original plaintext.

**Solution:**

(a) Because $J$ is 56 bits long, $C_2$ is 128-56 = 72 bits long. $C_1$ is a full 128-bit block and so the total length is 72+128 bits, same as the plaintext.

(b) To decrypt compute:

$$\begin{aligned} D(C_1, K) &= M_2 \parallel J \\ D(C_2 \parallel J, K) &= M_1 \end{aligned}$$

then discard $J$.

7. Suppose that the IV for CBC mode is chosen to be a counter instead of a random value. Show that an attacker can gain information regarding the first block of a ciphertext by using a chosen plaintext attack. More specifically, show that the attacker can check whether $C$ has first plaintext block equal to a particular block $P_0$ by asking for the ciphertext of a specific plaintext.

**Solution:** Suppose that the attacker observes a ciphertext $(IV, C_1, C_2, \ldots)$ which is the encryption of message $(P_1, P_2, \ldots)$. The attacker knows that the $IV$ used in the next ciphertext will be $IV+1$, so that it will be $(IV+1, C_1', C_2', \ldots)$. He can check whether the first plaintext block in the observed ciphertext is a particular value $\hat{P}$ by asking for the ciphertext of a message with first block equal to $\hat{P} \oplus IV \oplus (IV+1)$. The first block output in the new ciphertext will then be

$$C_1' = E((\hat{P} \oplus IV \oplus (IV+1)) \oplus (IV+1), K) = E(\hat{P} \oplus IV, K)$$

If the first plaintext block of the first observed message was indeed $\hat{P}$ then it follows that $C_1' = C_1$ which can be observed by the attacker.

8. Compare the following three modes of operation for block ciphers: ECB, CBC and CTR. Suppose that:

- AES is used as the block cipher;
- the padding method used is to append a single 1 at the end of the plaintext and then enough 0s to complete the block;
- the nonce used in counter mode has 96 bits.

How many ciphertext bits need to be transmitted in each of these three cases if the input plaintext has 104 bits?

**Solution:**

Note that we do *not* include sending the key. This must be set up securely before the transmission starts. We need to send the ciphertext and, where needed, the IV and nonce.

1. For ECB only the ciphertext block is sent, one block of 128 bits. The message must be padded before encryption and after decryption.
2. For CBC the ciphertext is also one block including padding, but we need to include 128 bits for the IV, so we obtain 256 bits in total.
3. For CTR there is no padding required since encryption in is like a stream cipher and we take only as many bits from the keystream as needed. So the basic encrypted messages needs exactly 104 bits. We also need to send the nonce of 96 bits. Thus in total we require 104 + 96 = 200 bits.

Note that these options are giving different security guarantees. ECB gives weak encryption, CBC and CTR give better encryption (randomized).

9. The CTR-DRBG described in the slides keeps a counter state $V$ and a key $K$ in memory. The pair $(K, V)$ is initialized, and occasionally updated, with a random seed using the Initialize and Reseed functions. In this question we assume that 128-bit keys and blocks are used (such as in AES) and also that each call to the Generate function outputs 10 blocks (1280 bits).

   Suppose that an attacker obtains the current $(K, V)$ value after 5 blocks have been generated during a call to Generate.

   (a) Explain why this attacker can obtain all of the 1280 bits generated from that call to Generate.

   (b) Show further that the attacker can obtain the output from all future calls to generate until Reseed is called.

   (c) Explain why this attacker cannot obtain the output from any previous calls to Generate, even if Reseed has never been called.

**Solution:**

(a) An attacker with a valid $(K, V)$ pair can generate all following outputs by incrementing $V$ and computing the block cipher output in the same way as the generator. The attacker can also decrement $V$ to compute $E(V, K)$ for previous values of the counter state $V$.

(b) At the end of the Generate function the Update function is called with an empty $D$ value. Thus given a valid $(K, V)$ pair the attacker can continue with the generation in the same way as the user.

(c) To obtain the output from earlier calls to Generate the attacker needs to find the previous $(K, V)$ pair in place before the Update function was called at the end of the previous Generate call. If $(K_0, V_0)$ was the old pair before the Update was called and $(K, V)$ the new pair after update was called then they are related by the following equation, where $inc$ indicates increment of the counter.

$$K \parallel V = (E(V_0, K_0) \parallel E(inc(V_0), K_0)) \oplus D$$

Since $D$ is empty when Update is called by Generate, and assuming that the block and key length are equal, we can write:

$$\begin{aligned} K &= E(V_0, K_0) \\ V &= E(inc(V_0), K_0) \end{aligned}$$

There is no reasonable way to solve these equations to find $K_0$ since all the attacker has is two ciphertexts from the unknown key $K_0$. Therefore it should be impossible to find the output from bits generated before the Update function was called. (This means that the Update function provides the *Backtracking resistance* property.)