Block Ciphers 2

Cryptology (1)

Martin Stanek

2025

KI FMFI UK Bratislava

Modes of operation

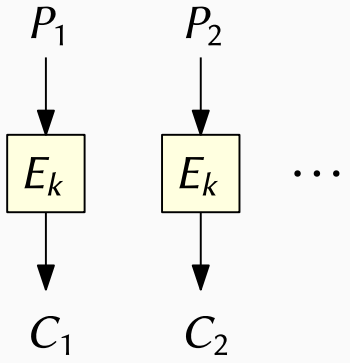
- plaintext usually much longer than the block length
- modes of operation can provide:
  - confidentiality (and not authenticity) – ECB, CBC, OFB, CFB
  - authenticity (and not confidentiality) – CMAC
  - confidentiality & authenticity (authenticated encryption) – GCM, CCM
  - confidentiality for block-oriented storage devices (disks) – XTS
  - key wrapping
  - format-preserving encryption, …
- varying requirements (speed, security properties, ability to parallelize, availability of RNG, etc.) ⇒ different modes for the same purpose

Confidentiality modes

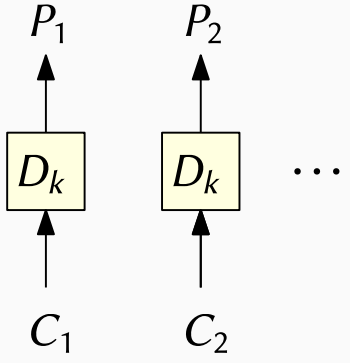
- the most important confidentiality modes: ECB, CBC, OFB, CFB, CTR
- e.g. see NIST SP 800-38A: *Recommendation for Block Cipher Modes of Operation: Methods and Techniques*
- None of these modes provide protection against accidental or adversarial modifications of the ciphertext!
- however, the effect of ciphertext modification on resulting plaintext varies among modes

ECB (Electronic Codebook)

- the simplest mode: $C_i = E_k(P_i)$, $P_i = D_k(C_i)$
- requires padding to ensure length that is a multiple of the block length
- encryption and decryption trivially parallelizable
- data leaks: $C_i = C_j \Leftrightarrow P_i = P_j$
- easy to rearrange the ciphertexts blocks (permute, duplicate, …)
- easy to perform a seek (random access)



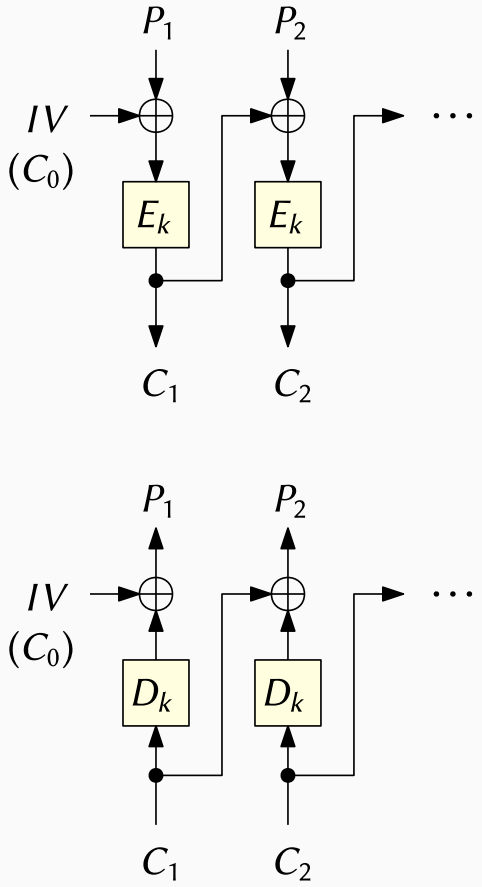
encrypt: $C_i = E_k(P_i)$



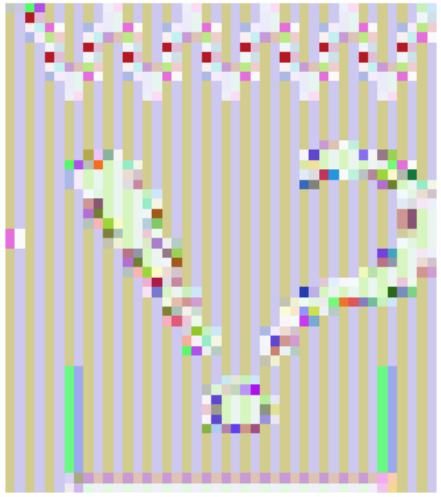
decrypt: $P_i = D_k(C_i)$

CBC (Cipher Block Chaining)

- encryption: $C_i = E_k(P_i \oplus C_{i-1})$
- decryption: $P_i = D_k(C_i) \oplus C_{i-1}$
- initialization vector IV – secrecy not required, usually appended as C_0
- popular mode (AES-128 CBC was mandatory in TLS 1.2)
- parallelizable decryption but not encryption
- similarly to ECB, plaintext should be a multiple of the block length
  - padding, ciphertext stealing



Visual comparison of ECB and CBC (AES-128)



ECB

CBC

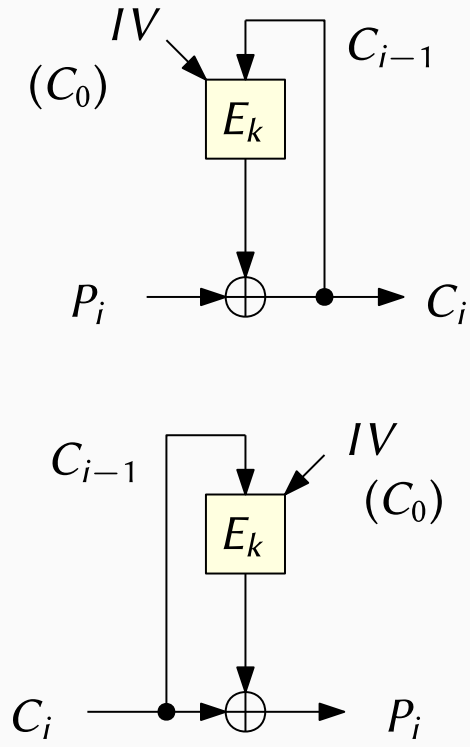
- IV should be unpredictable (e.g. IV $= E_k(\text{msg}_{\text{seq}})$, random, …)
  - otherwise, in CPA scenario, an attacker gets an $E_{k(\cdot)}$ oracle
- data leak (birthday & two-time pad):

$$C_i = C_j \quad \Rightarrow \quad E_k(P_i \oplus C_{i-1}) = E_k(P_j \oplus C_{j-1})$$
$$P_i \oplus P_j = C_{i-1} \oplus C_{j-1}$$

- Sweet32 attack (2016): ciphers with block length 64 bits and large amount of data encrypted using the same key (TLS, OpenVPN)
  - 64 bit block ⇒ collision expected after $\approx 2^{32}$ blocks (32 GB)
- limit number of blocks encrypted with a single key

CFB (Cipher Feedback)

- encryption: $C_i = P_i \oplus E_k(C_{i-1})$
- decryption: $P_i = C_i \oplus E_k(C_{i-1})$
- parallelizable decryption but not encryption
- D_k is not needed
- plaintext length does not need to be a multiple of the block length
- IV should be unique for each plaintext
  - repeated IV ⇒ two-time pad for the first blocks:
    $C_1 \oplus C_1' = E_k(\text{IV}) \oplus P_1 \oplus E_k(\text{IV}) \oplus P_1' = P_1 \oplus P_1'$



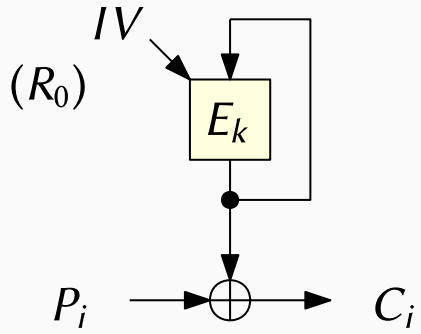
CFB8 variant of CFB mode and Zerologon

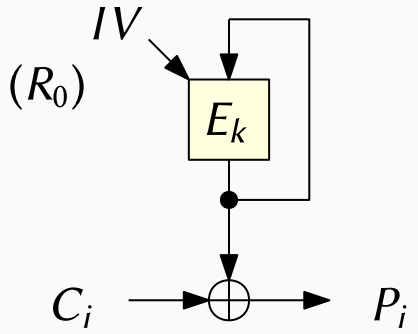
- Zerologon – compromising domain admin in AD (2020)
- problems with cryptography in Netlogon protocol (AES-CFB8)
- CFB8 mode (P_i and C_i are bytes):

$$C_1 = E_k(\mathrm{IV}[0...15])[0] \oplus P_1$$
$$C_2 = E_k(\mathrm{IV}[1...15]C_1)[0] \oplus P_2$$
$$C_3 = E_k(\mathrm{IV}[2...15]C_1C_2)[0] \oplus P_3$$
$$\ldots$$
$$C_{i+1} = E_k(C[i-15, \ldots, i])[0] \oplus P_{i+1}$$

- Netlogon implementation used all-zero IV (always)
  - consider all-zero plaintext
  - $1/256$ of all keys lead to all-zero ciphertext
- client authentication
  - encrypting his own challenge with a session key
- the attacker chooses all-zero challenge
  - session-key is unknown
  - succeess with probability $1/256$
  - repeat if necessary (session-key will change since it depends on the server challenge as well)

OFB (Output Feedback)





$$R_i = E_k(R_{i-1})$$

encrypt: $C_i = P_i \oplus R_i$ decrypt: $P_i = C_i \oplus R_i$

- synchronous stream cipher; D_k is not needed
- IV should be unique for each plaintext, otherwise we get two-time pad problem
- neither encryption nor decryption can be parallelized

CTR (Counter)

encrypt: $C_i = P_i \oplus E_k(IV \mid \text{ctr})$ decrypt: $P_i = C_i \oplus E_k(IV \mid \text{ctr})$

- inputs to E_k should not overlap (otherwise ... two-time pad)
- similar to OFB (synchronous stream cipher)
- easy to perform a seek (random access)
- easy to encrypt and decrypt in parallel

Padding

- ECB and CBC assume that n divides the length of the plaintext
- padding required (various paddings are used):
  - bit padding – append 1 (always) and necessary number of zeroes: msg || 1000…0
  - byte padding (PKCS #7, CMS (RFC 5652)):

$$\begin{aligned} &\text{msg} \,\|\, 01 && \text{if } n \mid |\text{msg}| + 1 \\ &\text{msg} \,\|\, 03\ 03\ 03 && \text{if } n \mid |\text{msg}| + 3 \\ &\text{msg} \,\|\, 01\ 01\ \ldots\ 01 && \text{if } n \mid |\text{msg}| \quad (\text{for } n = 128) \end{aligned}$$

  - similarly for TLS 1.2 (RFC 5246): 00; 02 02 02; 0F 0F … 0F
- padding $\Rightarrow$ |ciphertext| $>$ |plaintext|
- padding should be verified after decryption
- “stream” modes like OFB, CTR or CFB do not need padding, |ciphertext| $=$ |plaintext|

Padding oracle attack

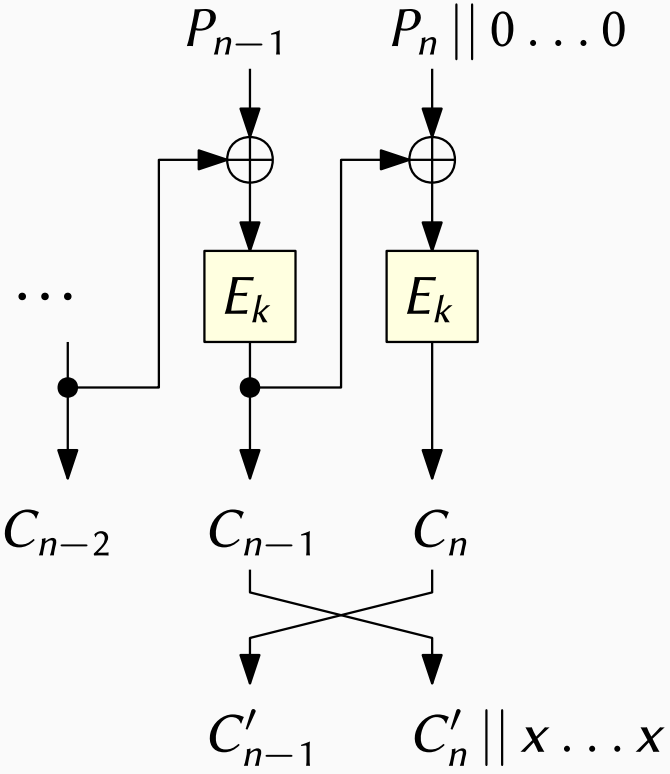
- implementation issue
- our assumptions:
  - CBC mode, PKCS 7 padding
  - we can recognize correct/incorrect padding, e.g., a server behaves differently (observable error, timing differences, …)
- goal: decrypt a ciphertext block C, i.e., compute $Y = D_k(C)$
- the attack:
  - let X be a random 15-byte block
  - try ciphertexts: $(X \parallel 00) \parallel C$, $(X \parallel 01) \parallel C$, …, $(X \parallel 7A) \parallel C$, …, $(X \parallel FF) \parallel C$, until we find the ciphertext with valid padding
  - the highest probability: the corresponding plaintext ends with byte 01 (and not with bytes 02 02 or even longer padding)

Padding oracle attack (cont.)

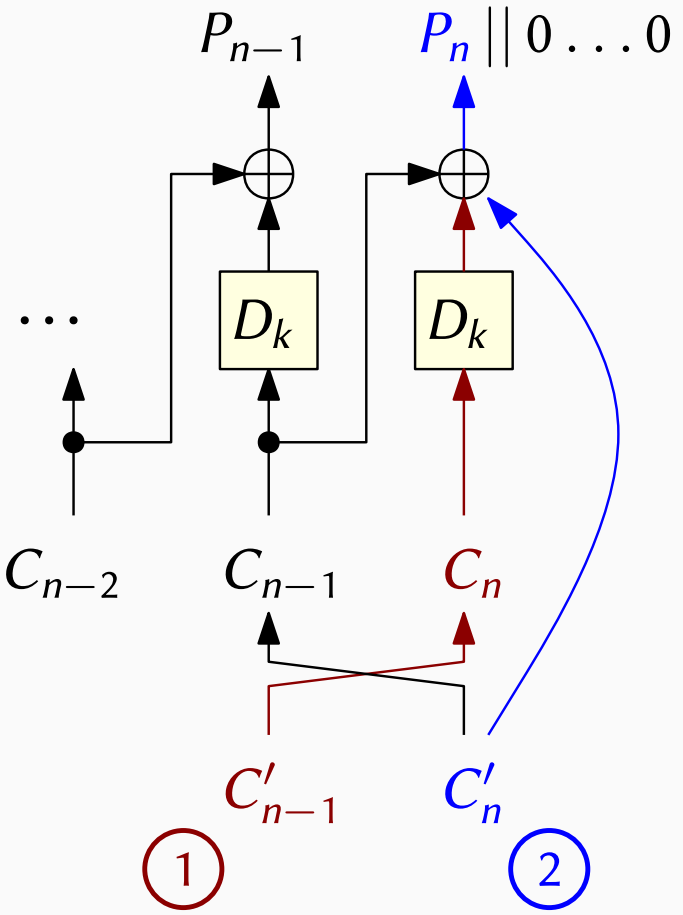
- the attack (cont.):
  - there is always a candidate with 01 padding, we can also alter the penultimate byte of X to distinguish it
  - finally, we can compute Y_{15}, e.g., $7\text{A} \oplus Y_{15} = 01 \Rightarrow Y_{15} = 7\text{B}$
  - set the last byte of the first block to get 02 as the final byte of the plaintext: $b \oplus Y_{15} = 02 \Rightarrow b = 79$
  - try ciphertexts (X is a random 14-byte value): $(X \parallel 00 \parallel 79) \parallel C$, $(X \parallel 01 \parallel 79) \parallel C$, …, $(X \parallel \text{B2} \parallel 79) \parallel C$, …, $(X \parallel \text{FF} \parallel 79) \parallel C$, until we find a ciphertext with valid padding (this time: 02 02)
  - we can compute Y_{14}, e.g., $\text{B2} \oplus Y_{14} = 02 \Rightarrow Y_{14} = \text{B0}$
  - … similarly for other bytes
- a variant used against SSL/TLS implementations (Lucky Thirteen, 2013)

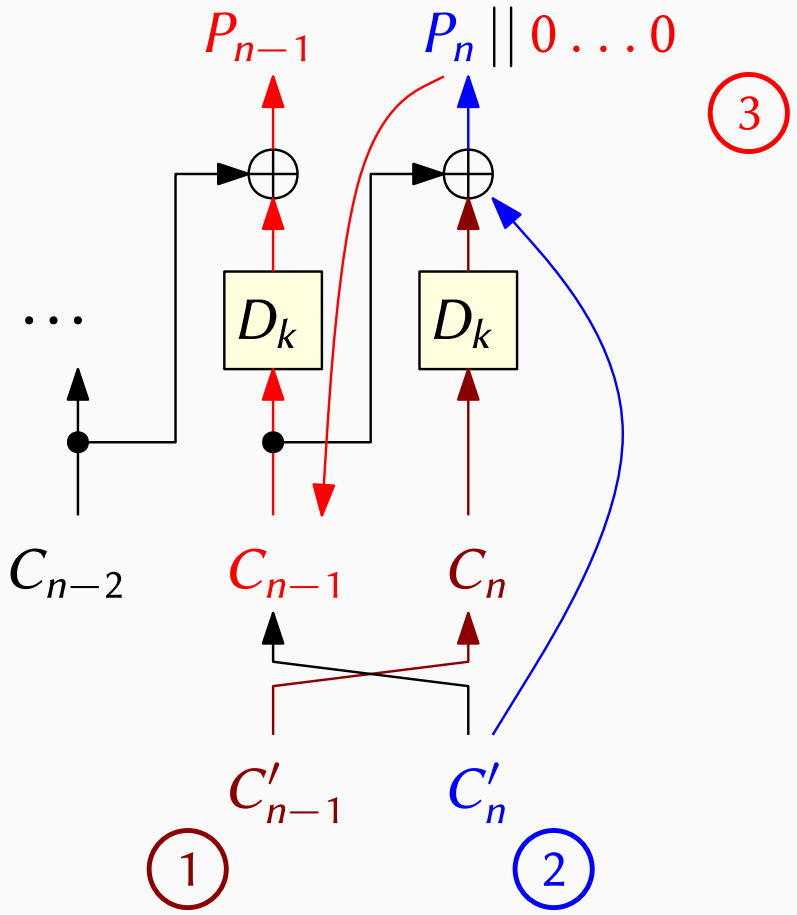
Ciphertext stealing 1

- method of avoiding padding for CBC or ECB modes
- ciphertext stealing for CBC mode encryption
  - example: Kerberos, AES256-CTS
- plaintext: $\ldots P_{n-2}, P_{n-1}, P_n$
- ciphertext: $\ldots C_{n-2}, C'_{n-1}, C'_n$



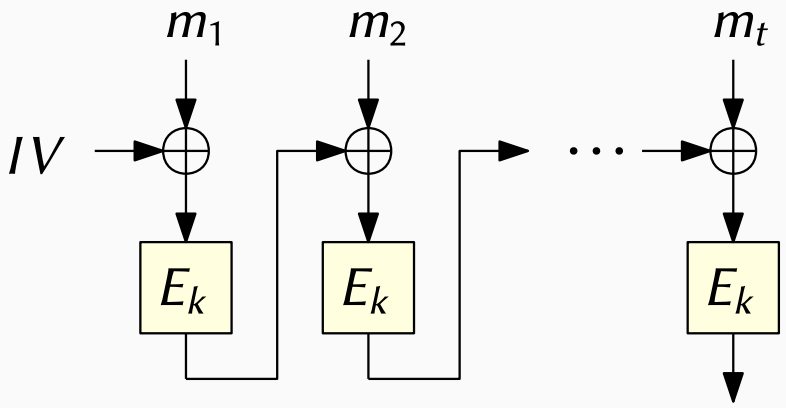
Decrypting CBC ciphertext stealing





CBC-MAC

- using a block cipher for authentication
- MAC – Message Authentication Code
  - secret key + message (data) ↦ authentication tag
  - sender computes and sends the authentication tag
  - recipient recomputes the tag and compares with the received value



- secure for fixed-length messages
- security discussion in the MAC lecture
  - insecure for variable-length messages
  - key and IV sensitivity

Authenticated encryption

- modes providing confidentiality & authenticity of data
- examples: CCM (Counter with CBC-MAC), GCM (Galois/Counter Mode)
- CCM (idea):
  - plaintext encrypted using CTR mode
  - authentication tag computed as CBC-MAC
  - authenticate-then-encrypt (single key is used)
  - two-pass scheme (E is used twice for each input block)

Authenticated encryption – GCM

- NIST SP 800-38D, GCM for 128-bit block ciphers, such as AES
  - popular variant with 96-bit IV and 32-bit counter

Notation:
- K – key, single key is used
- P, A, C – plaintext, additional authenticated data, ciphertext
- $H = E_K(0^{128})$ – *authentication key* used for authentication tag computation
- $J_0 = \text{IV} \parallel 0^{31} \parallel 1$;
- $\text{len}(X)$ – the length of X in bits, a 64-bit value

Encryption using the CTR mode

1. $\text{ctr} = \text{inc}_{32}(J_0)$
   increment the last 4B modulo 2^{32}
2. $P \mapsto X_1, \ldots, X_n$
   the last block might be incomplete
3. for $i = 1, \ldots, n$:
   $C_i = P_i \oplus E_K(\text{ctr})$
   $\text{ctr} = \text{inc}_{32}(\text{ctr})$
4. output: $C_1, \ldots, C_n$, where $|C_n| = |P_n|$

$\mathbf{GHASH}_H(A, C)$:

1. $A \parallel C \mapsto X_1, \ldots, X_{n-1}, \underbrace{\text{len}(A) \parallel \text{len}(C)}_{X_n}$

   A and C are padded with 0 to fill incomplete blocks, if necessary

2. $Y_0 = 0^{128}$
3. for $i = 1, \ldots, n$: $Y_i = (Y_{i-1} \oplus X_i) \bullet H$
4. $\text{GHASH}_H(A, C) \leftarrow Y_n$

– authentication tag T: $T = E_K(J_0) \oplus \text{GHASH}_H(A, C)$

– $\bullet$ is multiplication in $\text{GF}(2^{128})$, the field is generated by $x^{128} + x^7 + x^2 + x + 1$

GCM remarks and forbidden attack

- limited message length (increasing the length affects the security), for example: (TLS 1.3, RFC 8446) *For AES-GCM, up to* $2^{24.5}$ *full-size records (about* 24 *million) may be encrypted on a given connection while keeping a safety margin of approximately* 2^{-57} *for Authenticated Encryption (AE) security.*
- IV must be unique (*nonce*) for given key and message, otherwise *forbidden attack*
- repeated IV:
  - two-time pad for CTR encryption
  - H can be computed (see next slides)
  - impact: the attacker can manipulate ciphertext (bit flipping), edit associated data A, and compute correct authentication tag
- this implementation issue was observed in real world systems in the past

Forbidden attack – let's compute H

- *forbidden attack* (A. Joux)
- assumption: two messages encrypted with the same K and IV
- H is the same in both cases, since $H = E_K(0^{128})$
- similarly $E_K(J_0)$ is the same (let's denote it J^*)
- for readability: $\oplus \mapsto +$ and $\bullet \mapsto \cdot$
- computation of T can be written as a polynomial $g(z)$:

$$g(z) = J^* + z \cdot X_n + z^2 \cdot X_{n-1} + \ldots + z^n \cdot X_1$$

  where $T = g(H)$
- known: T, A, C, where $A \parallel C \mapsto X_1, \ldots, X_{n-1}, \text{len}(A) \parallel \text{len}(C)$
- unknown: H and J^*

Forbidden attack – let's compute H (cont.)

- two polynomials for our messages:

$$g(z) = J^* + z \cdot X_n + z^2 \cdot X_{n-1} + \ldots + z^n \cdot X_1$$

$$g'(z) = J^* + z \cdot X'_{n'} + z^2 \cdot X'_{n'-1} + \ldots + z^{n'} \cdot X'_1$$

- H is a root of $g(z) + T$ and $g'(z) + T' \Rightarrow$ it is a root of their sum: $g(z) + T + g'(z) + T'$
  - polynomial with degree $\max\{n, n'\}$, we know all coefficients (J^* cancels out)
- H can be computed via factorization, finding roots and verification for other messages
  - more messages with the same IV $\Rightarrow$ more polynomials that share a common root
  - number of roots in theory up to the degree, in practice substantially less

Exercises

1. *Analyze how inverting a bit in the ciphertext changes the resulting plaintext after decryption. Consider ECB, CBC, OFB, CFB, and CTR modes.*
2. *Assume CPA scenario for the CBC mode with predictable* IV. *Show how an attacker can get an access to* $E_k(\cdot)$ *oracle. Discuss how this allows to test the candidates for the plaintext block (for given ciphertext block).*

   (*) Show similar problem for a constant IV in the CFB mode.
3. *Assume a plaintext consisting of a sufficiently long sequence of 32-bit integers* $\langle 0 \rangle_{32}$, $\langle 1 \rangle_{32}$, $\langle 2 \rangle_{32}$, *... A block cipher with 32-bit block is used to encrypt this plaintext. Can you recognize which one of these modes was used: ECB, CBC, or OFB?*