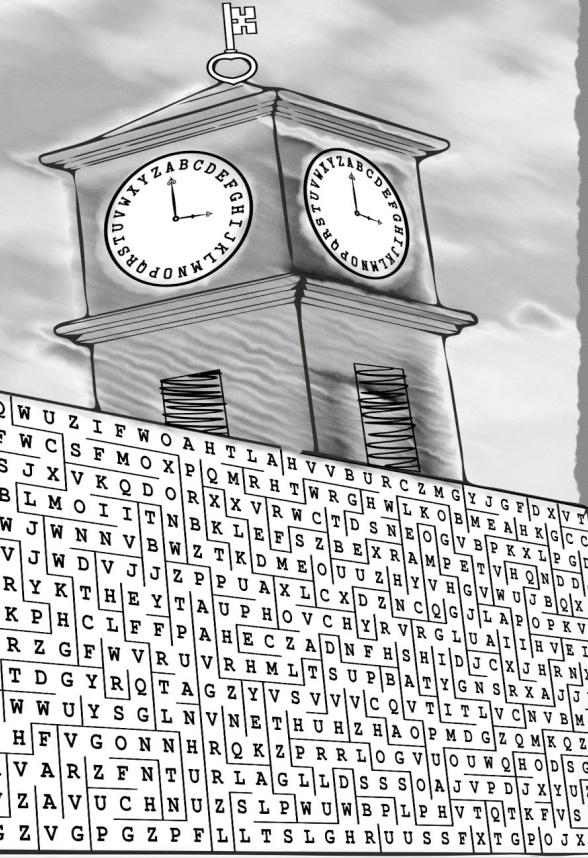


3

Szimmetrikus kulcsú rendszerek

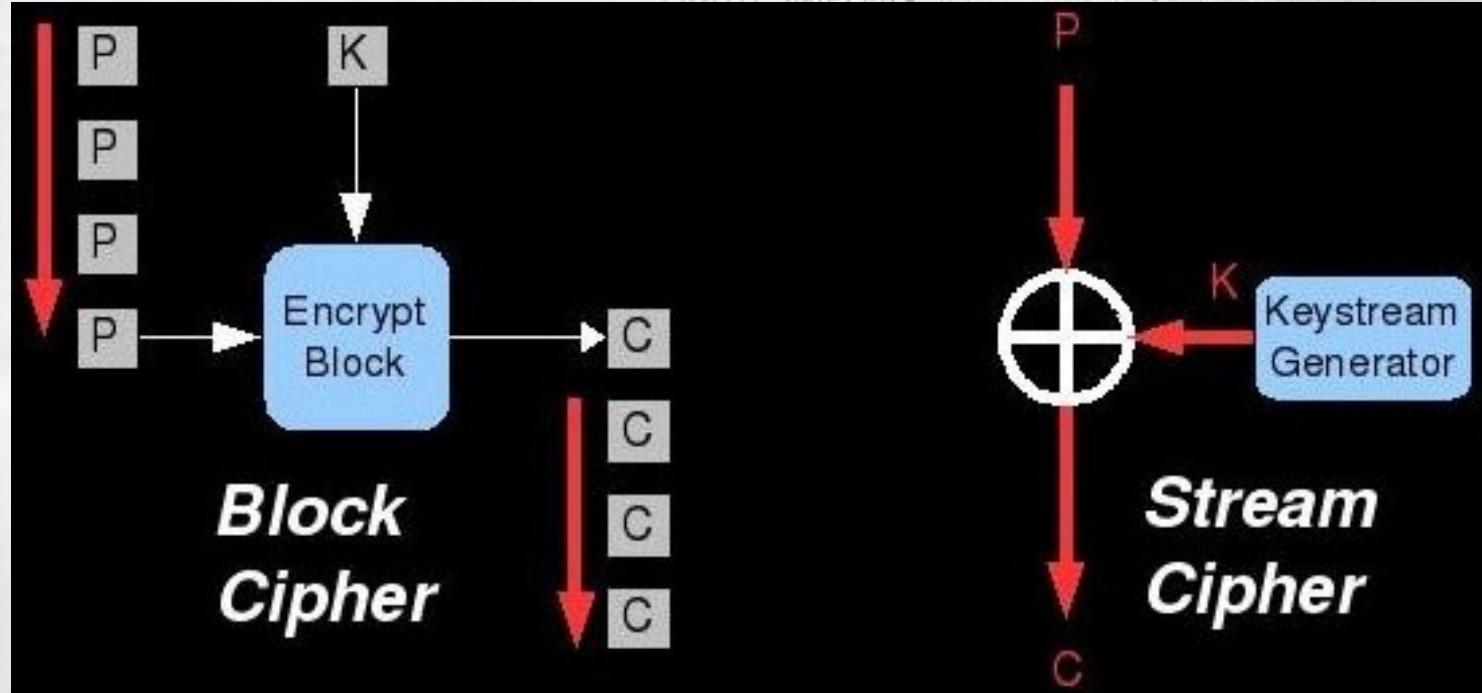
Modern rendszerek



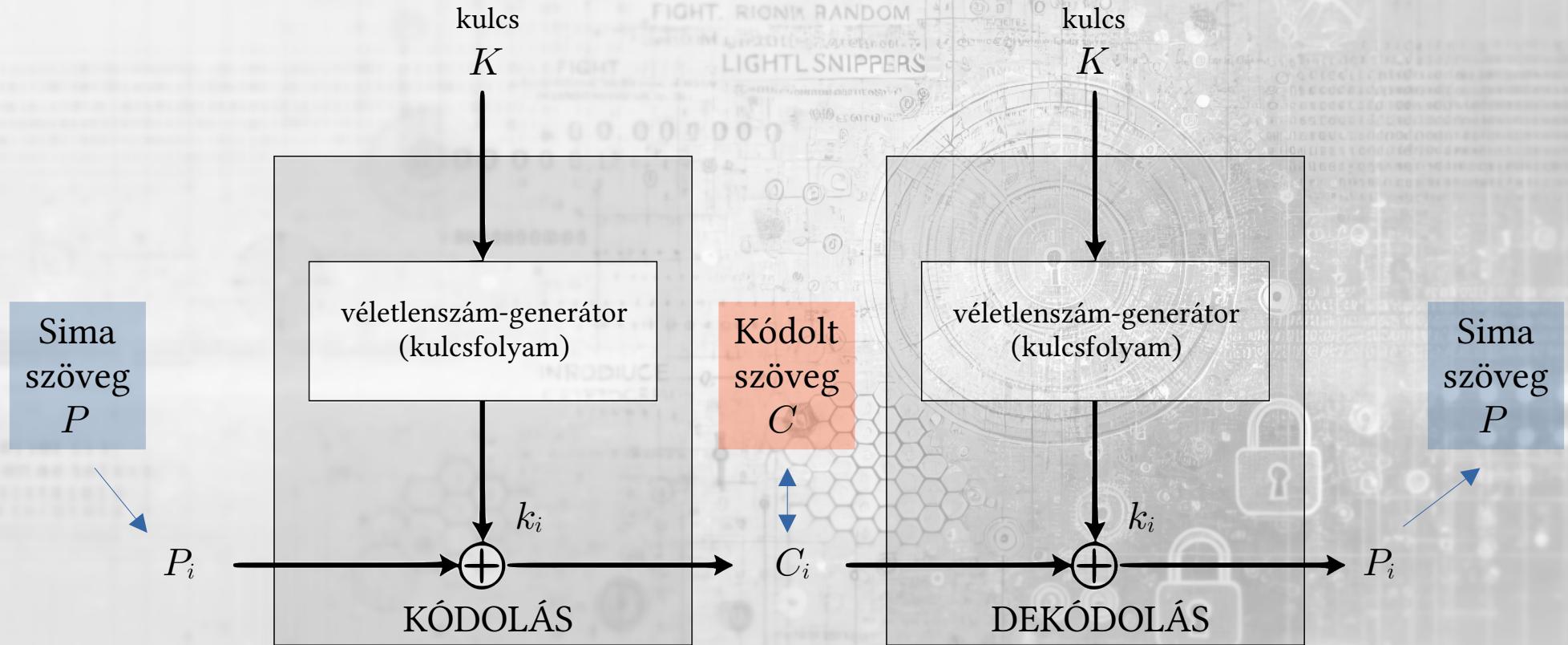
Modern titkosító rendszerek

- A modern rendszerek két osztályba sorolhatók:
 1. **Folyamtitkosítók** – az eredeti szöveg egységeit (betűit, bitjeit vagy bájtjait) egyenként titkosítják (pl. A5/1, RC4, véletlen átkulcsolás, stb.)
 2. **Tömbtitkosítók** – adott méretű adattömböt titkosítanak (pl. DES, IDEA, AES, stb.)

A modern rendszerek fajtái



A folyamtitkosítók általános felépítése



Véletlen átkulcsolás (one time pad)

A módszer lényege az, hogy az eredeti szöveget szövegegenségenként (karakterenként, betűnként, bitenként) összeadjuk modulo n a szöveggel azonos hosszúságú kulccsal. Itt n a használt karakterek száma:

- angol ábécé esetében $n = 26$
- bitek használata esetében $n = 2$ (tehát az összeadás modulo 2 az XOR bináris művelet lesz)

Véletlen átkulcsolás (one time pad)

Kulcs: Az eredeti szöveggel azonos méretű, lehetőleg véletlenszerű szövegegységsorozat, amelyet **csak egyszer** használunk fel (innen az angol elnevezés). Az ilyen kulcsot még *kulcsfolyamnak* is nevezzük. Jelöljük k_i -vel a k kulcs i -edik egységét.

Kódolás: $E_k(P) = C$, $C_i = (P_i + k_i) \text{ mod } n$, ahol P_i az eredeti szöveg i -edik egysége, C_i pedig a titkosított szöveg i -edik egysége.

Dekódolás: $D_k(C) = P$, $P_i = (C_i - k_i) \text{ mod } n$.

Véletlen átkulcsolás (one time pad)

1. példa. Angol ábécé esetében ($n = 26$):

$$\begin{aligned} P &= \text{T I T K O S} = 19 \ 8 \ 19 \ 10 \ 14 \ 18 \\ k &= \text{A C Z D P Q} = 0 \ 2 \ 25 \ 3 \ 15 \ 16 \\ C &= \frac{\text{T K S N D I}}{\text{P}} = \frac{19 \ 10 \ 18 \ 13}{0} \ 3 \ 8 \end{aligned} + (\text{mod } 26)$$

2. példa. A szövegegyeségek bitek ($n = 2$):

$$\begin{aligned} P &= 0 \ 1 \ 1 \ 0 \ 1 \ 0 \ 0 \ 0 \ 1 \ 0 \ 1 \ 1 \ 1 \\ k &= \text{1 0 1 0 0 1 0 0 0 1 1 0 0} \\ C &= \frac{0 \ 1 \ 1 \ 0 \ 0 \ 1 \ 1 \ 0 \ 0 \ 1 \ 1 \ 0 \ 1 \ 1}{1 \ 0 \ 1 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 1 \ 1 \ 0 \ 0} \end{aligned} + (\text{mod } 2)$$

Véletlen átkulcsolás (one time pad)

- Kriptoanalízis:
 - véletlen kulcsok egyszeri használatával **tökéletes titkosítást** valósít meg (C. Shannon, 1940)
 - sebezhetővé válik a rendszer, ha a kulcsot többször használjuk fel vagy ha a kulcsfolyam nem véletlen(szerű)
- Problémát jelenthet a kulcskezelés/generálás

A Solitaire algoritmus

- az algoritmust Neal Stephenson *Cryptonomicon* című regényéhez készítette Bruce Schneier (a regényben Pontifex néven szerepel)
- egy „kézi” használatra tervezett, véletlenszerű számsorozatot generáló módszer
- ötvözni kell véletlen átkulcsolással (a Solitaire egy kulcsfolyam-generáló módszer)
- használatához egy 54 lapos francia kártyapakli szükséges: a pókerben használatos 52 kártyalap és két (megkülönböztetett) dzsóker.

A Solitaire algoritmus

- Kulcs: a kártyapakli kezdeti állapota
- Az egyes kártyalapok értéke (bridzs sorrend):

A	2	3	4	5	6	7	8	9	10	J	Q	K	A	2	3	4	5	6	7	8	9	10	J	Q	K
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26

A	2	3	4	5	6	7	8	9	10	J	Q	K	A	2	3	4	5	6	7	8	9	10	J	Q	K
27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52

53 53

A Solitaire algoritmus

A módszer lépései:

1. Kezünkbe vesszük a megkevert kártyapaklit. Megkeressük benne a fehér dzsókert, és kicseréljük az alatta levő kártyalappal. Ha a fehér dzsóker a pakli legalsó kártyalapja, akkor beszúrjuk felülről az első kártyalap után.
2. Megkeressük a fekete dzsókert, és két kártyalappal lentebb szúrjuk be a pakliba. Ha a fekete dzsóker a legalsó kártyalap, akkor felülről a második kártyalap után szúrjuk be. Ha a fekete dzsóker a pakli alján az utolsó előtti kártyalap, akkor a legfelső kártyalap alá kerül.

A Solitaire algoritmus

3. Cseréljük fel az első dzsóker előtti kártyalapokat a második dzsóker utáni kártyalapokkal. Ennél a lépésnél nem számít a dzsókerek színe, csak az, hogy a kártyapakli tetejétől számítva melyik az első illetve a második dzsóker.
4. Megnézzük a legalsó kártyalapot, majd felülről annyi kártyalapot számlálunk, amennyi a legalsó kártyalap számértéke majd itt „törjük” a paklit úgy, hogy a legalsó kártya a helyén marad. Ha az utolsó kártyalap dzsóker, akkor a pakli nem módosul ennél a lépésnél.

A Solitaire algoritmus

5. Felülről annyi kártyalapot számlálunk le, amennyi a legfelső kártyalap számértéke, és megnézzük, hogy mi a következő kártyalap. Ennek a kártyalapnak a számértéke lesz a kulcsfolyam eleme. Ez a lépés nem módosítja a kártyapakliban levő lapok sorrendjét. Ha az első kártyalap éppen egy dzsóker, akkor nem számolunk és nem írunk ki semmit, hanem az első lépéssel folytatjuk.

1. és 2. lépések

A Solitaire algoritmus (példák)

Eredeti sorrend:



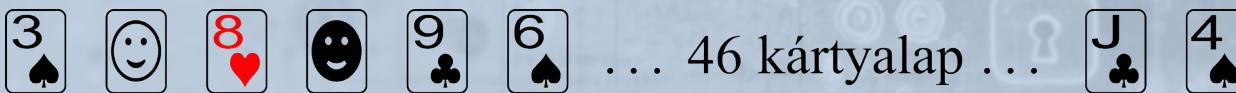
Az első két lépés elvégzése után:



Eredeti sorrend:

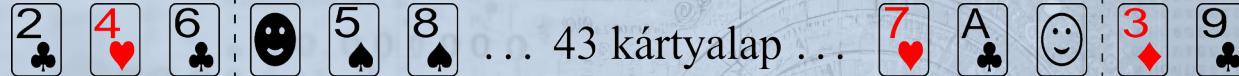


Az első két lépés elvégzése után:



3. lépés

Eredeti sorrend:



A 3. lépés elvégzése után:



Eredeti sorrend:



A 3. lépés elvégzése után:



4. lépés

A Solitaire algoritmus (példák)

Eredeti sorrend:



A 4. lépés elvégzése után:



- Ha az utolsó kártyalap dzsóker, akkor a pakli nem módosul ennél a lépésnél.
- Vigyázni kell arra, hogy a megcserélt részeken belül a kártyalapok sorrendje ne változzon meg.

5. lépés

Pakli állapota:



... 44 kártyalap ...



Mivel a legfelső kártyalap a , aminek a számértéke 50, pontosan ennyi kártyalapot számolunk a pakliban (az ötvenedik lap a), majd a következő lap számértékét kiírjuk egy papírra. Az ötvenedik kártyalap után következő lap a , aminek az értéke 27. Ez a szám lesz tehát a kulcsfolyam eleme.

- A pakli nem módosul ennél a lépésnél. Ha az első kártyalap éppen egy dzsóker, akkor az első lépéssel folytatjuk.

Blum–Blum–Shub-algoritmus

- Generálunk két nagy p, q prímet úgy, hogy $p, q \equiv 3 \pmod{4}$.
- $n = pq$.
- Legyen $s \in \{1, \dots, n - 1\}$ véletlen
- $x_0 := s^2 \bmod n$
- $x_i := x_{i-1}^2 \bmod n$ és $z_i := x_i \bmod 2$ (vagyis x_i paritása)
- Az álvéletlen bitsorozat: z_1, z_2, z_3, \dots

Megj.: Rögzített p és q mellett az s tekinthető a folyamtitkosító kulcsának.

Tömbtitkosítók

- rögzített méretű szövegegységtömböket titkosítanak (legtöbbször bittömböket)
- ha az eredeti üzenet hosszabb, mint a tömb rögzített mérete, akkor valamilyen **működési mód**ot használunk
- iterált tömbtitkosító: többször ismétlődik ugyanaz az f menetfüggvény, iterált bemenetekkel

Iterált tömbtitkosítók

- Legyen n a menetek száma és $P_i = f(P_{i-1}, k_i)$
 - $P = P_0$ az eredeti üzenet
 - $C = P_n$ a kódolt üzenet
 - k_i a k kulcsból generált i -edik segédkulcs
- Kulcsfehérítés: $P_0 = P \oplus k_0$ és $C = P_n \oplus k_{n+1}$

A kitöltés (padding)

- az eredeti üzenet kiegészítése bizonyos számú bittel úgy, hogy az üzenet bithossza a rögzített tömbméret egész számú többszöröse legyen
- tipikus kitöltési módok:
 - 0 értékű bitekkel pótolunk;
 - (DES módszer): 1 értékű bit, majd 0 értékű bitek;
 - (Schneier, Ferguson): n darab n értékű bajttal egészítünk ki

Működési módok

A NIST négy működési módot vezetett be 1980-ban ([FIPS 81](#)-es szabvány), ezt később kiegészítették a CTR móddal:

1. ECB (Electronic codebook) - Elektronikus kódkönyv mód
2. CBC (Cipher-block chaining) - Rejtjeles blokkok láncolása
3. CFB (Cipher feedback) - A titkos szöveg visszacsatolása
4. OFB (Output feedback) - A kimenet visszacsatolása
5. CTR (Counter) – Számláló mód

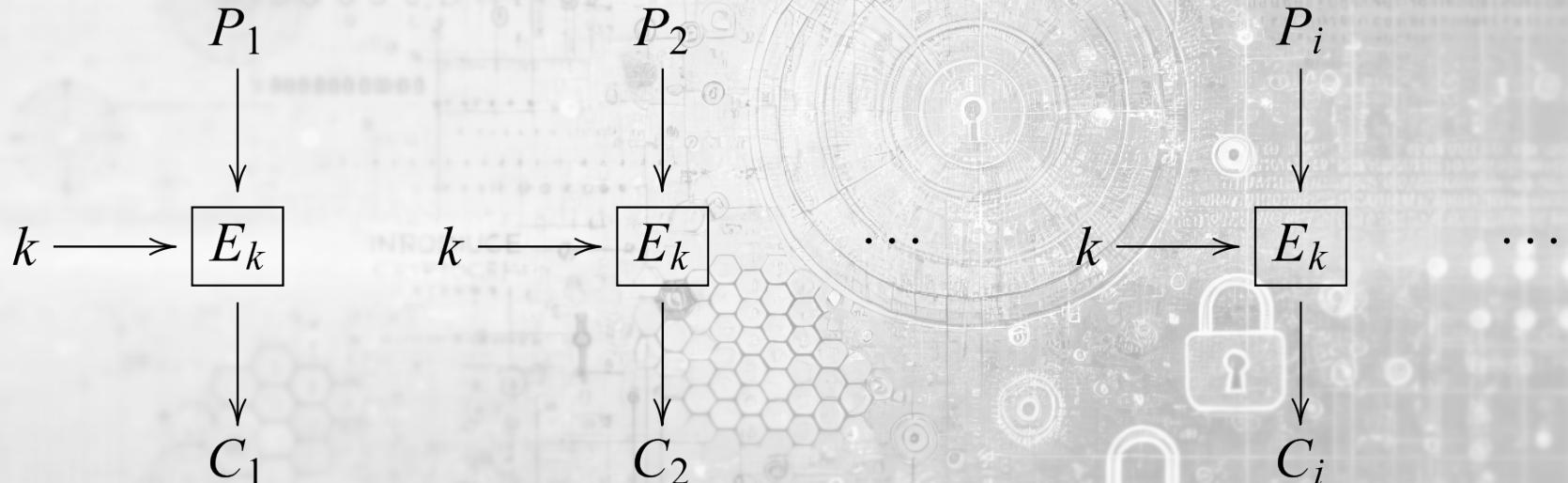
Működési módok

- Jelölések:

- P_i - az eredeti üzenet i -edik tömbje
- C_i - a kódolt üzenet i -edik tömbje
- k - kulcs
- E_k - titkosító eljárás
- D_k - dekódoló eljárás
- \oplus - az XOR művelet
- IV - kezdeti tömb
- \parallel - bittömbök egymás után való illesztése (konkatenálás)

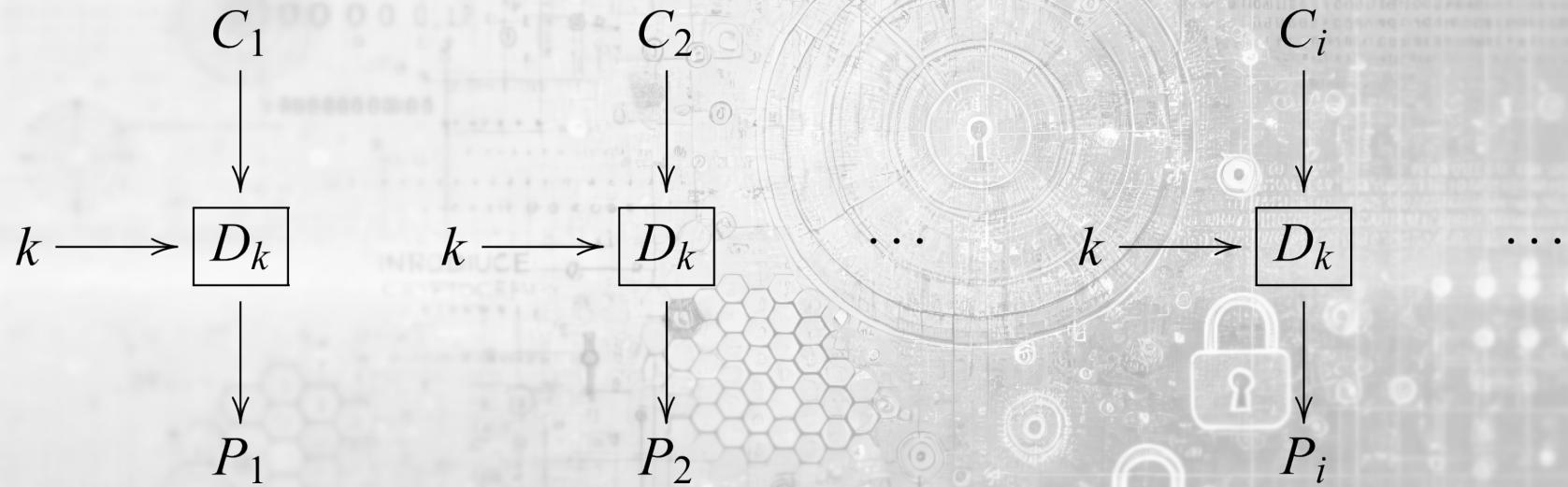
ECB mód (Electronic codebook mode)

Kódolás: $C_i = E_k(P_i)$, minden i tömbre.



ECB mód (Electronic codebook mode)

Dekódolás: $P_i = D_k(C_i)$, minden i tömbre.

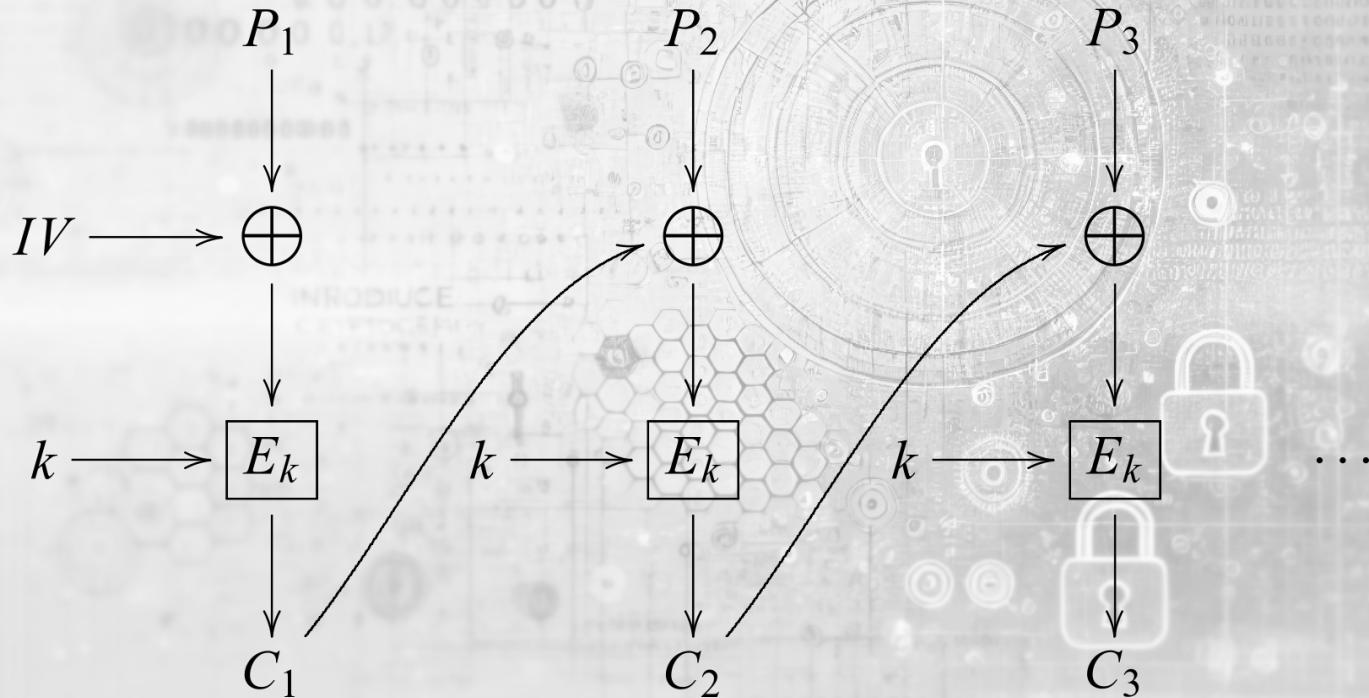


ECB mód (Electronic codebook mode)

- Identikus tömböket identikus tömbökbe kódol (ha $P_i = P_j$, akkor $C_i = C_j$), mint egy óriási kódkönyv.
- **Előnyök:** egyszerű használat, párhuzamosítható
- **Hátrányok:**
 - a blokkokat egymástól függetlenül kódolja
 - a sima szöveg ismétlődő blokkjai ismétlődnek a titkosított szövegen is (ha az eredeti üzenet strukturált, akkor ez kihasználható)

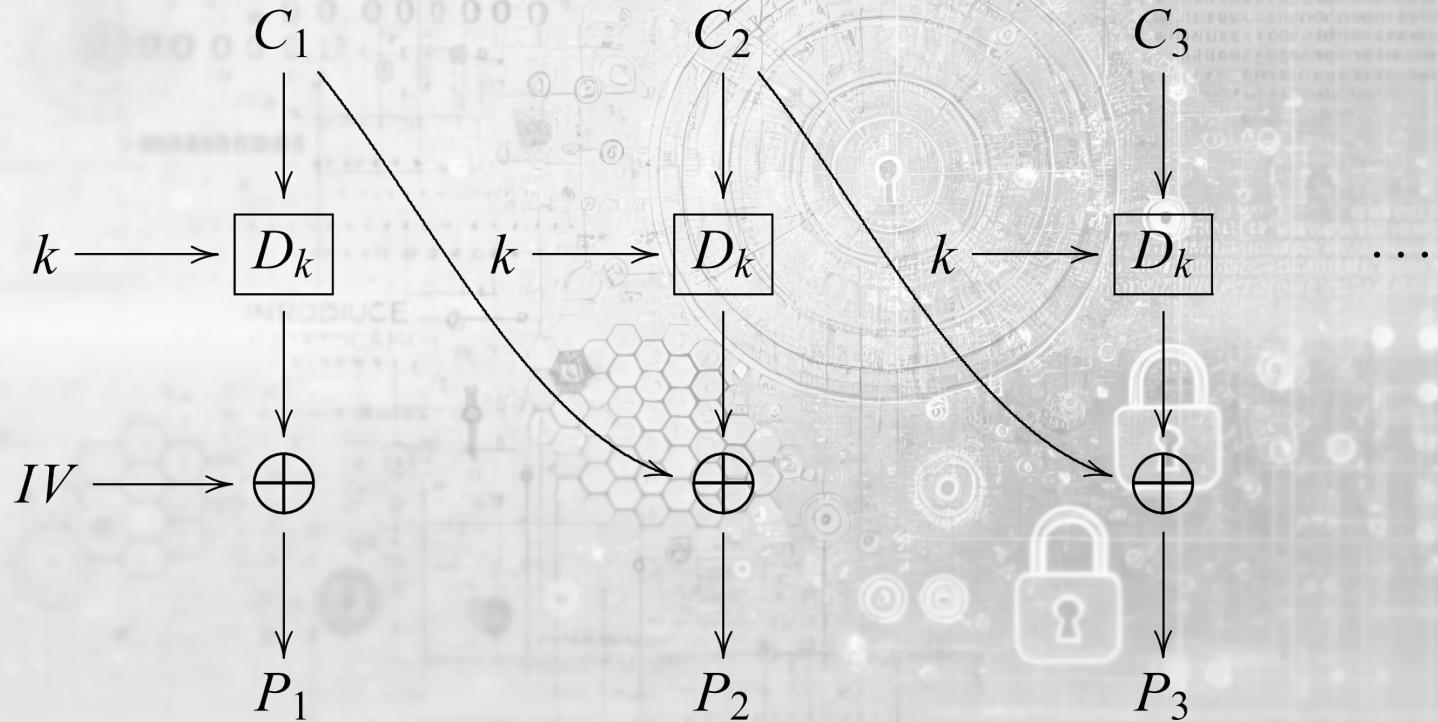
CBC mód (Cipher-block chaining mode)

Kódolás: $C_0 = IV$, minden további C_i tömbre: $C_i = E_k(P_i \oplus C_{i-1})$, $\forall i > 0$.



CBC mód (Cipher-block chaining mode)

Dekódolás: $P_i = D_k(C_i) \oplus C_{i-1}$, $C_0 = IV$.



CBC mód (Cipher-block chaining mode)

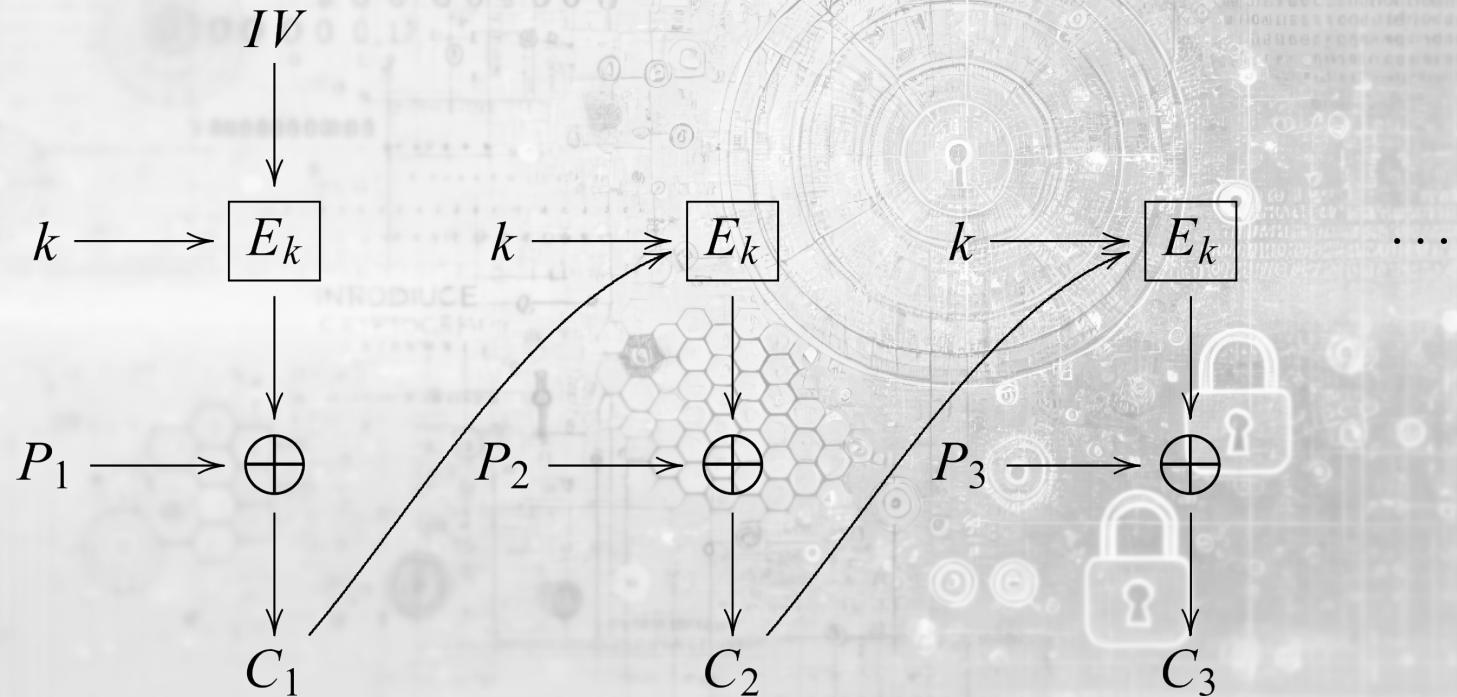
- A CBC az egyik legelterjedtebb működési mód.
- Egy bit változtatása a P_i tömbben megváltoztatja a C_j tömböket minden $j \geq i$ -re.
- Egy bit változtatása a C_i tömbben megváltoztatja a P_i tömböt és egy darab bitet a P_{i+1} tömbben.

CBC mód (Cipher-block chaining mode)

- Szükség van egy *IV* kezdeti tömbre (ezt a küldő és a fogadó fél is ismeri):
 - ha *IV* nem titkos, akkor rögzíteni kell az értékét (mert ha kódolás nélkül küldik át, akkor a támadó megváltoztathatja)
 - egyébként titkosítani kell a kezdeti tömböt is (pl. ECB móddal)
- **Előnyök:** egy titkosított blokk minden előtte levőtől függ, a dekódolás párhuzamosítható
- **Hátrányok:** a kódolás nem párhuzamosítható

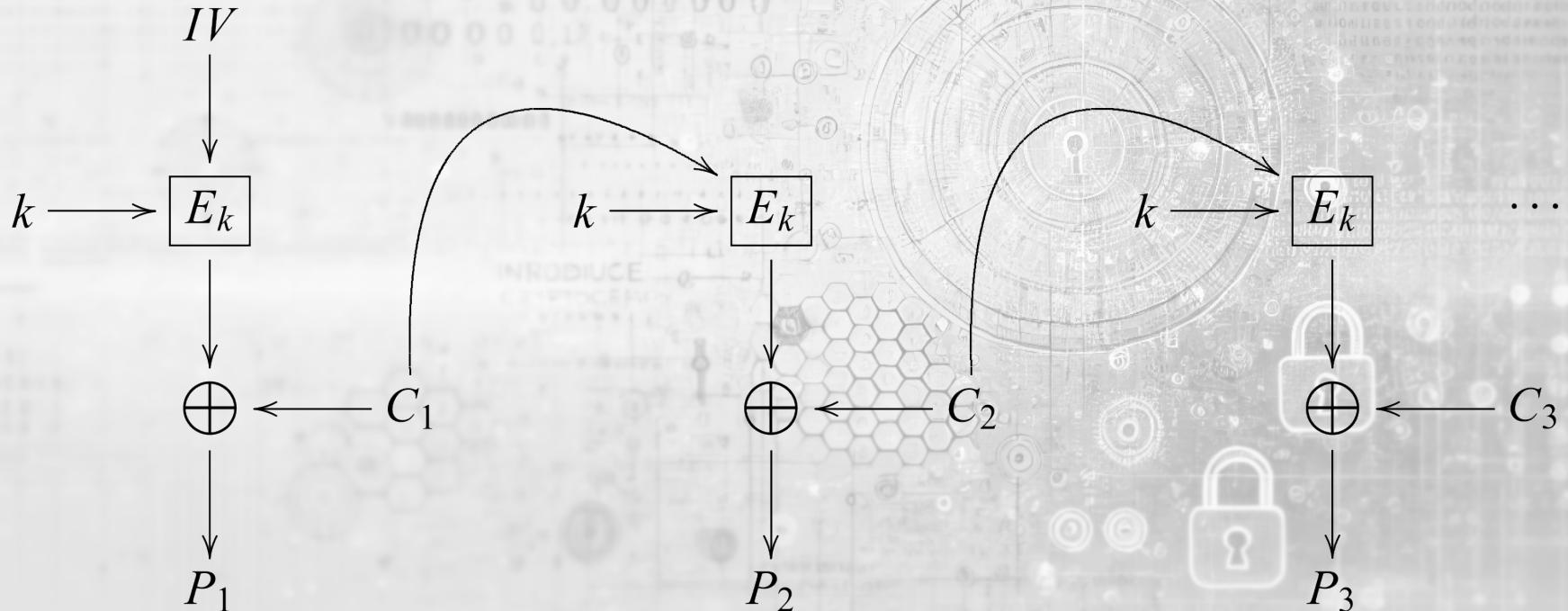
CFB mód (Cipher feedback mode)

Kódolás: $C_i = E_k(C_{i-1}) \oplus P_i$, $C_0 = IV$.



CFB mód (Cipher feedback mode)

Dekódolás: $P_i = E_k(C_{i-1}) \oplus C_i, C_0 = IV.$



CFB mód (Cipher feedback mode)

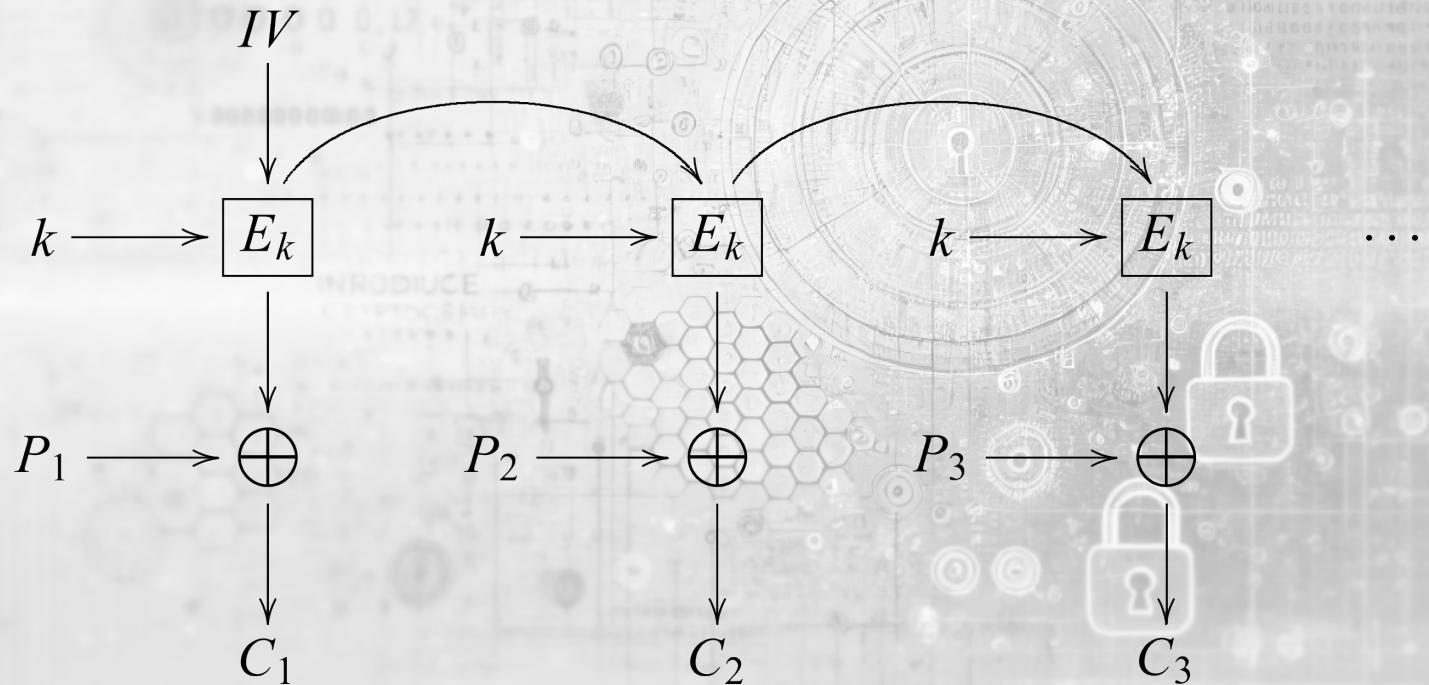
- A CFB mód a tömbtitkosítót átalakítja egy folyamtitkosítóvá.
- Egy bit változtatása a P_i tömbben megváltoztatja a C_j tömböket minden $j \geq i$ -re.
- Egy bit változtatása a C_i tömbben megváltoztatja a P_{i+1} tömböt és egy darab bitet a P_i tömbben.

CFB mód (Cipher feedback mode)

- Csak a kódoló eljárást használja, a dekódolót nem.
- **Előnyök:** a dekódolás párhuzamosítható
- **Hátrányok:**
 - a kódolás nem párhuzamosítható
 - érzékenyebb az adatátviteli hibákra

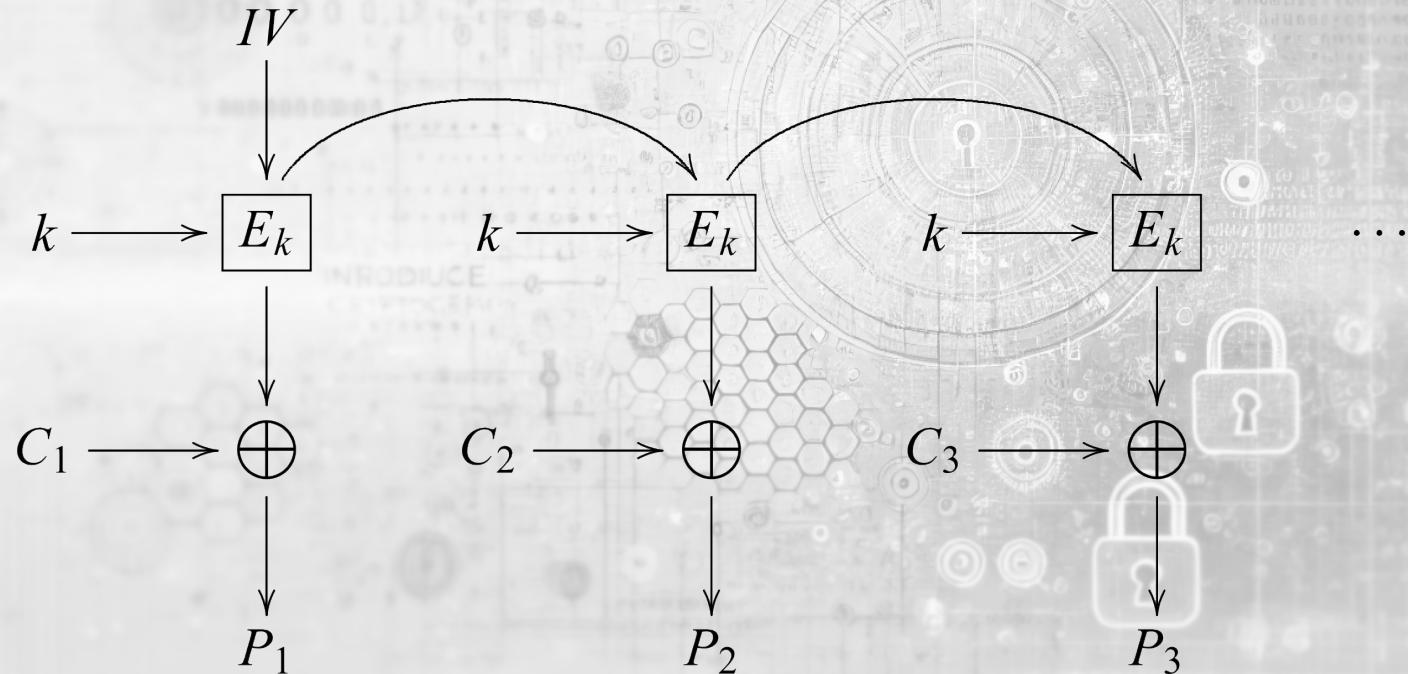
OFB mód (Output feedback mode)

Kódolás: $C_i = P_i \oplus O_i$, $O_i = E_k(O_{i-1})$, $O_0 = IV$.



OFB mód (Output feedback mode)

Dekódolás: $P_i = C_i \oplus O_i$, $O_i = E_k(O_{i-1})$, $O_0 = IV$.

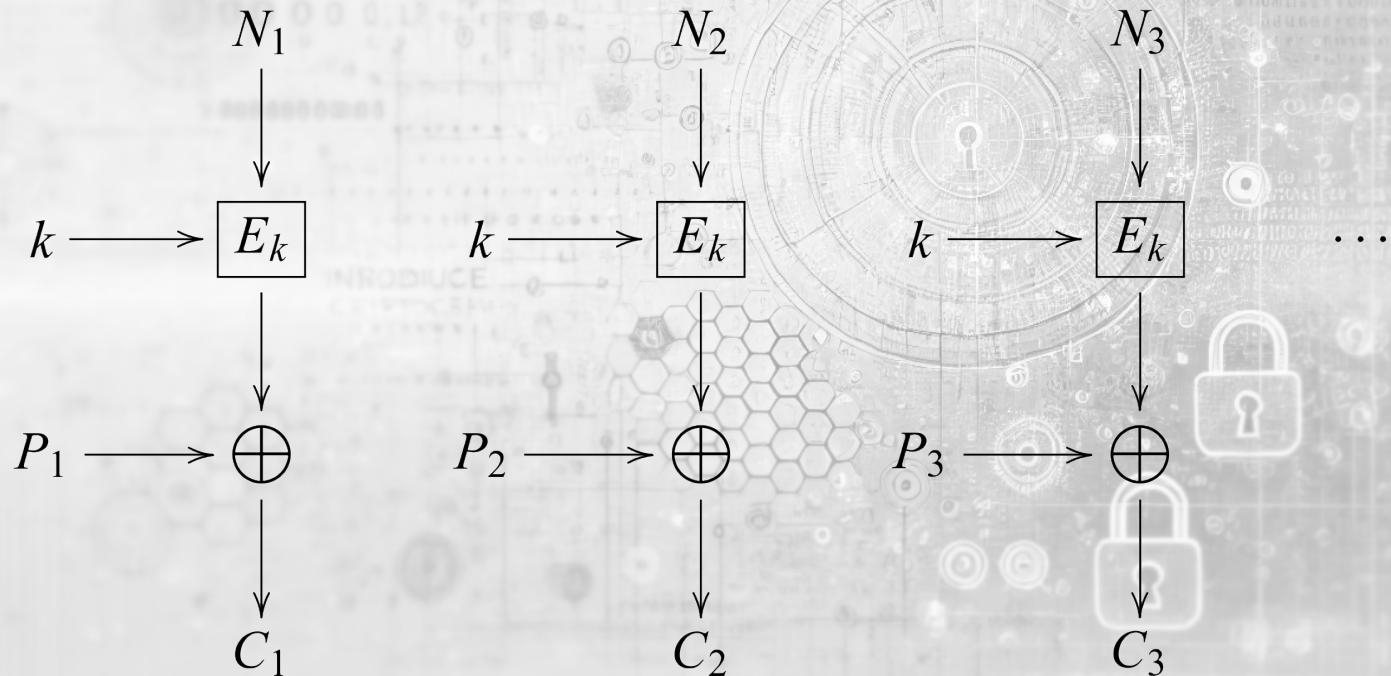


OFB mód (Output feedback mode)

- A OFB mód a tömbtitkosítót átalakítja egy folyamtitkosítóvá.
- Egy bit változtatása P_i -ben (vagy C_i -ben) pontosan egy bitet változtat C_i -ben (vagy P_i -ben).
- **Előnyök:** a kódolás/dekódolás párhuzamosítható (ha előre legeneráljuk az O_i sorozatot)
- **Hátrányok:** érzékenyebb az adatátviteli hibákra

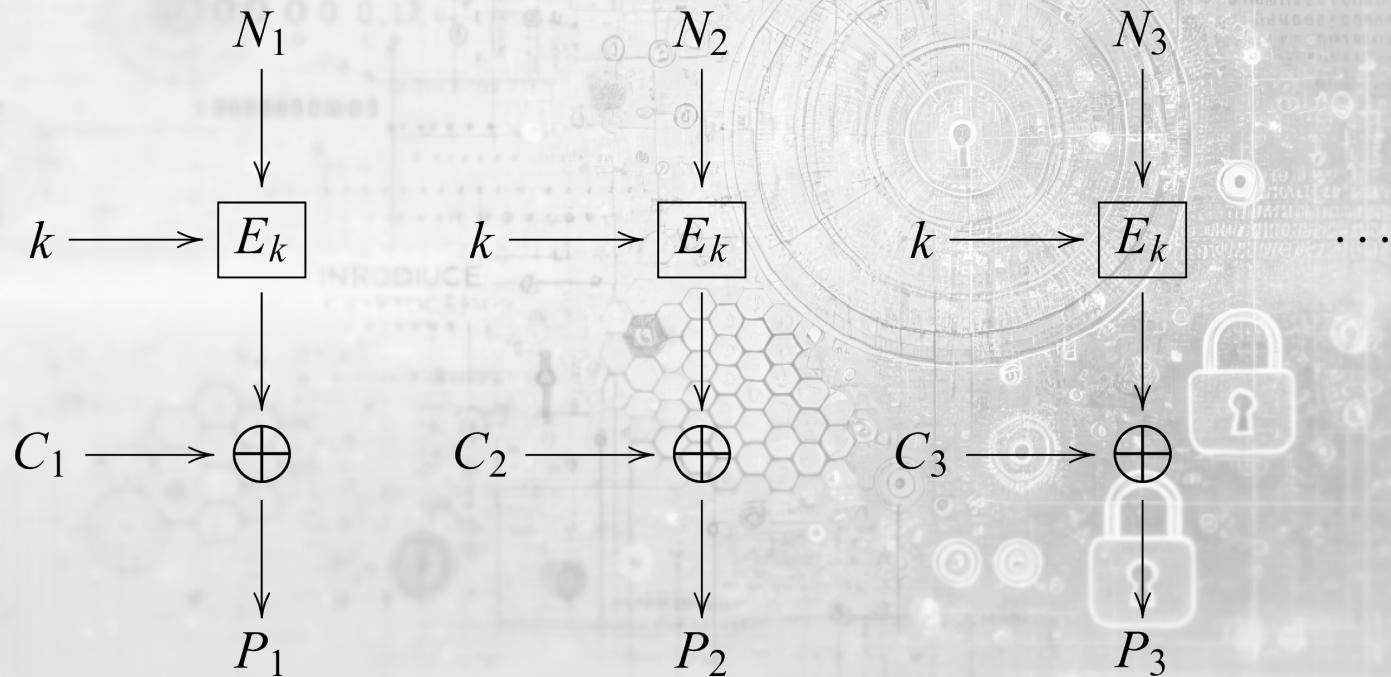
CTR mód (Counter mode)

Kódolás: $C_i = P_i \oplus E_k(N_i)$, ahol $N_i = IV \parallel \underbrace{00\dots0}_{\text{számláló}}$.



CTR mód (Counter mode)

Dekódolás: $P_i = C_i \oplus E_k(N_i)$, ahol $N_i = IV \parallel \underbrace{00\dots0}_{\text{számláló}} i$.



CTR mód (Counter mode)

- A CTR mód nagy előnye, hogy alkalmas párhuzamos kódolásra és dekódolásra, tehát nagyon gyors.
- Csak a kódoló eljárást használja.
- Közvetlenül is hozzáférhetünk az adatblokkokhoz (random access).
- Vigyáznunk kell, hogy az IV kezdeti vektort ne használjuk újra ugyanazzal a kulccsal.

A lavina-effektus

- a lavina-effektus (vagy lavinahatás) azt jelenti, hogy kis változtatás az eredeti üzenetben vagy kulcsban a kimenet drasztikus változását eredményezi
- ha a bemenet vagy kulcsnak csak egy bitjét változtatjuk meg, ez a kimeneti bitek körülbelül felének a megváltozásával járjon
- ha csak kismértékű lenne a változás, akkor az lehetőséget adna a nyílt szöveg vagy a kulcs keresési terének leszűkítésére
- általában szükséges, hogy a tömbtitkosítók már néhány menet után is megvalósítsák a lavina-effektust

Általános támadások a tömbtitkosítók ellen

1. differenciális kriptoanalízis

- azt vizsgáljuk, hogy a bemenetek közötti különbségek hogyan befolyásolják a kimeneti különbséget
- rögzített a különbségeket követünk egy transzformációs hálózaton keresztül és ez felfedheti, hogy a titkosító hol mutat nem véletlenszerű viselkedést, ezáltal csökkentve a kulcsteret

Általános támadások a tömbtitkosítók ellen

2. lineáris kriptoanalízis

- nyílt szöveg ismeretén alapuló támadás
- a lineáris kriptoanalízis két fontos részből áll:
 - olyan lineáris összefüggések felírása az eredeti szöveg, a kódolt szöveg és a kulcs bitjei között, amelyek nagy (1-hez közel) valószínűsséggel igazak, vagy nagy valószínűsséggel nem teljesülnek.
 - ezen lineáris összefüggések felhasználása arra, hogy nyílt- és a nekik megfelelő kódolt szövegek (szövegpárok) ismerete alapján meghatározzuk a kulcs bitjeit