usepackagetikzfeynman

Információs rendszerek biztonságtechnikája

Kriptográfia
Szimmetrikus kódú algoritmusok

Vakulya Gergely

A kriprográfia négy célja

Titkosság (secrecy, confidentality)

Annak biztosítása, hogy az üzenetet harmadik fél **ne tudja elolvasni**.

Hitelesség (authentication)

Annak bizonyíthatósága, hogy az üzenet valóban a **feladótól származik**.

Letagadhatatlanság (nonrepudiation)

Annak bizonyíthatósága, hogy az üzenetet a **feladó valóban elküldte**.

Sértetlenség (integrity)

Annak biztosítása, hogy az üzenetet harmadik személy **ne tudja megváltoztatni**.

A kriptográfia modellje

Jelölések

- Nyílt szöveg (plaintext), P
- Kulcs (key), K
- Titkosított szöveg (cyphertext), C

$$C = E_K(P)$$

$$P = D_K(C)$$

$$D_K(E_K(P)) = P$$

Kerchoff elve

Minden algoritmusnak nyilvánosnak kell lennie; csak a kulcsok titkosak (1883)

Kriptoanalízis

Csak titkosított szöveg alapú támadás

A kódtöréshez csak egy, vagy több titkosított szöveget ismerünk. A legnehezebb szituáció.

Ismert nyílt szöveg alapú támadás

Nyílt szöveg - titkosított szöveg párokat ismerünk. Talán ez a leggyakoribb eset.

Választott nyílt szöveg alapú támadás

Lehetőségünk van tetszőleges nyílt szöveget titkosítani (tehát a nyílt szövegekhez előállítani azok titkosított szöveg párját), de magát a kulcsot nem ismerjük. Ez az eset nyilvános kulcsú algoritmusok, illetve „blackbox” titkosítók esetében fordul elő.

Az ismeretlenség biztonsága (security by obscurity)

A rendszer (nem feltétlenül titkosítási eljárás) működésével kapcsolatban bizonyos gyengeségeket szándékosan nem hoznak nyilvánosságra azt remélve, hogy azokat nem fedezik fel / használják ki. Napellenzőbe „rejtett” forgalmi engedély, lábtörlő alá „rejtett” kulcs esete.

- Zárt forráskódú firmware-ek.
- A fájlrendszer mélyére „jól elrejtett” fájlokban tárolt érzékeny adatok.
- Beépített kiskapuk (például pótjelszavak).

Rossz gyakorlat. Kerülendő!

A feltörhetetlen kód (one-time pad)

A kizáró vagy (XOR) művelet

- $K \oplus K = O$
- $C = P \oplus K$
- $P = C \oplus K = P \oplus K \oplus K$
- Egy kulcs csak egyszer használható fel!

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

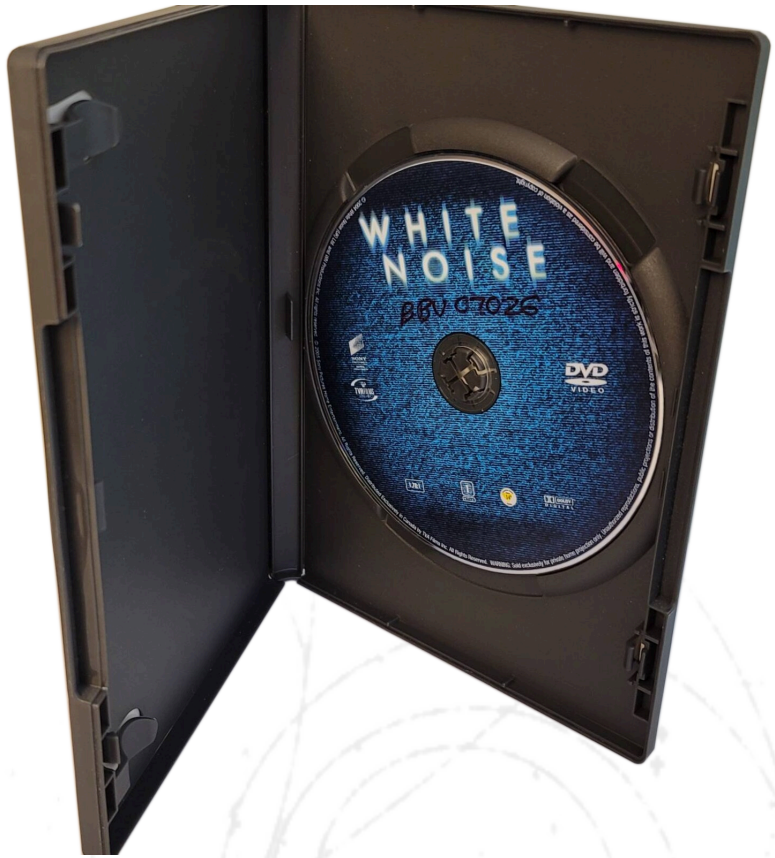
A titkosítás menete

- Alice és Bob megállapodnak egy közös kulcsban.
- A kódolás akkor működőképes, ha a kulcs véletlenszerű bitekből áll és legalább akkora, mint a kódolandó adat.
- Például Alice generál egy DVD-nyi véletlenszerű bitet és a lemezt eljuttatja Bobnak.
- Ezt követően bármelyikük titkos kommunikációt tud folytatni a másikkal a kulcs (a DVD) méretéig bezárólag.

A feltörhetetlen kód (one-time pad)

Hátrányok

- A kulcs elkészítése problematikus.
  - A számítógép általában pszeudorandom szekvenciákat tud gyorsan generálni.
  - Ténylegesen véletlen számok előállítása nehéz / lassú.
- A kulcs megosztása, tárolása problematikus.
  - Elzárva kell tartani.
  - Könnyen lemásolhatják.

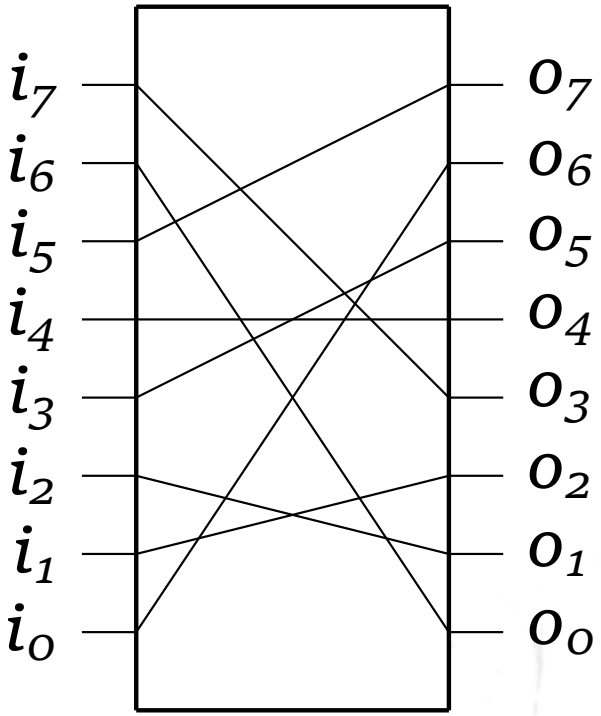


Szimmetrikus kulcsú titkosítások

- A szimmetrikus kulcsú algoritmusok mind a kódoláshoz, mind a dekódoláshoz **ugyanazt a kulcsot** használják.
- A blokk kódolók n bites blokkokra bontják a bemenetet és a kódolást / dekódolást ilyen egységenként végzik.
- Kívánatos, hogy egy k bites nyílt szöveg egy szintén k bites titkosított szöveget eredményezzen, viszont k nem mindig egész számú többszöröse n-nek. Megoldás: Az utolsó blokk feltöltése, **padding**.
- A vevő oldalnak tudnia kell, hogy a használt kulcs helyes-e. Megoldás: **redundancia** alkalmazása (pl. ellenőrző összeg).

P-doboz

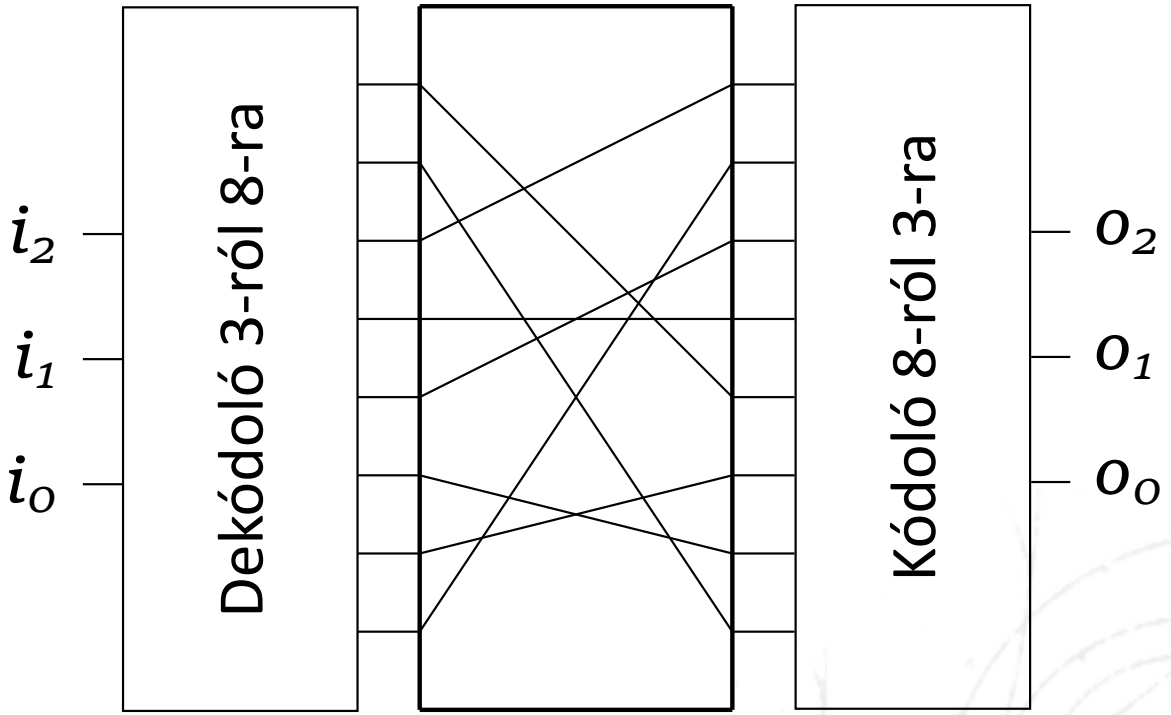
- P: Permutate
- A bemenet bitjein keverést végez



- Lényegében egy keverő kódoló.

S-doboz

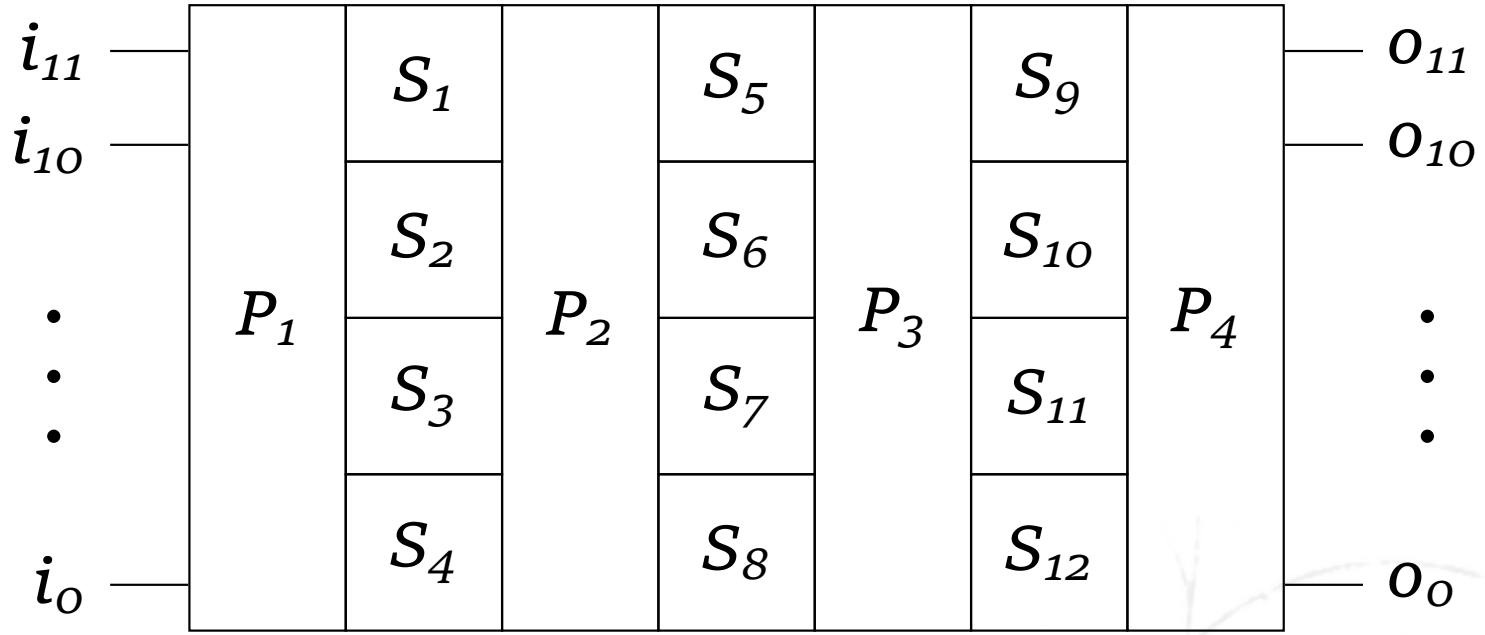
- S: Substitution
- Minden bitmintát egy másik bitmintára cserél



- Az S-doboznak része egy P-doboz is.
- Lényegében egy helyettesítő kódoló.

Data Encryption Standard (DES)

- A P- és S-dobozokból szorzattitkosítók építhetők



- Az egyes lépésekben felváltva keverés és kicserélés történik.
- A P- és S-dobozok, valamint a szorzattitkosítók **reverzibilisek**.

Data Encryption Standard (DES)

A DES háttere

- Az IBM fejlesztette ki 1977-ben.
- Többek közt P és S dobozokat is felhasználó szorzat típusú kódoló 16 iterációs lépéssel.
- Kezdetben 128 bites, majd a végleges szabványban 56 bites kulcs.
- Kezdettől fogva számos kritika övezi.

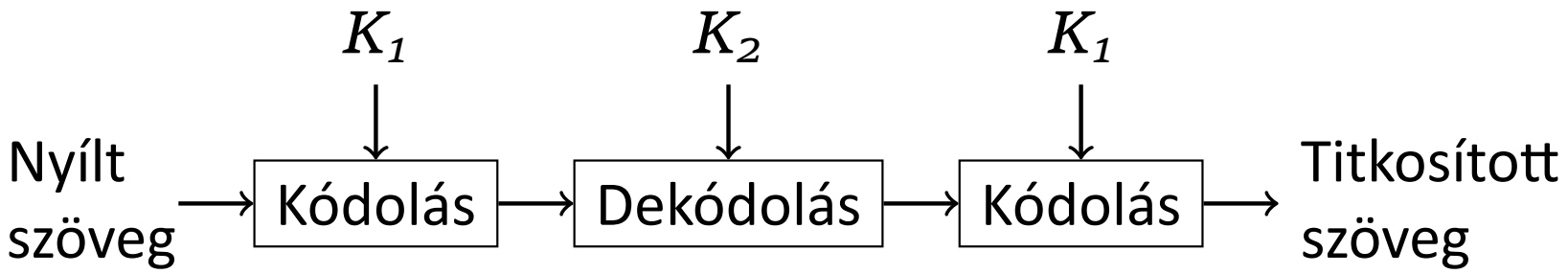
Próbálkozások a DES feltörésére

- Az 56 bites kulcs 2^{56} ($\approx 10^{15}$, 72 billiárd) nagyságú kulcsteret ad.
- A DES megalkotása után néhány évvel több különböző törési kihívás (néhány hónaptól néhány hétig tartó idők).
- 2006, „COPACOBANA”: Egy 120 db FPGA-ból álló kompakt rendszer, egy napon belüli törési lehetőség.

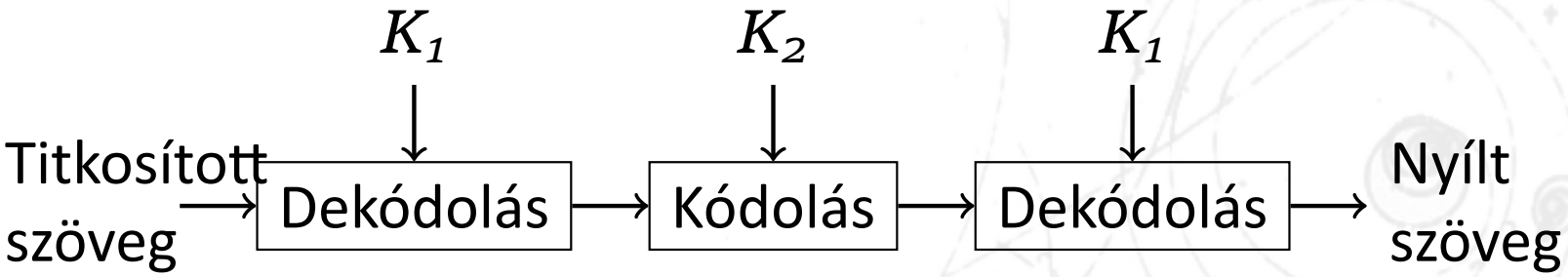
Tripe-DES (3DES)

- A DES kiterjesztése (1979).
- Három DES lépést végez el egymás után.
- 112 bites kulcsot használ.

A titkosítás menete



A visszafejtés menete



A 3DES trükkje

- A 3DES 3 lépésben titkosít (és dekódol), de csak két kulcsot használ.
- Az első és az utolsó kulcs azonos.
- Ráadásul a második lépésben fordított művelet (kódolás helyett dekódolás és fordítva) történik.
- Gondoljuk át a $K_1 = K_2$ esetet!
- Ekkor ugyanaz történik, mintha egyszer titkosítanánk a hagyományos DES-sel.
- **Kompatibilitás!**

Az AES háttere

- A DES nem tűnt kellően megbízhatónak.
- A gyenge 56 bites kulcs mellett a DES-t számos összeesküvés is övezte.
- Szükség volt egy független, erős, modern titkosítási eljárásra.
- Kriptográfiai verseny (1997). Célja *egy nyílt, publikusan nyilvánosságra hozott titkosító algoritmus, ami képes megvédeni az érzékeny kormányzati információkat bőven a következő évszázadig*.
- Legfontosabb követelmények:
  - Szimmetrikus blokk-kódoló
  - Támogassa a 128, 192 és 256 bites kulcsokat
- A verseny győztese fogja majd az AES (Advanced Encryption Standard) szerepét betölteni.

A kriptográfiai verseny kritériumai

A verseny kiírásakor három csoportban foglalták össze a kiértékelés kritériumait:

- Biztonságra vonatkozó kritériumokat (a kiértékelés legfontosabb szempontjai):
  - Az adott titkosító algoritmus biztonsága összevetve a többi pályázóval
  - Milyen mértékben tér el az algoritmus kimenete a bemeneti blokk véletlenszerű permutációjától
  - Az algoritmus biztonságának matematikai megalapozottsága
- Költségek
  - Licenszköltség (az AES-nek jogdíjmentesen elérhetőnek kell lennie)
  - Számításigény
  - Memóriaigény
- Algoritmikus és implementációs tulajdonságok
  - Flexibilitás (például újabb kulcsméretek támogatása)
  - Hardveres és szoftveres is megvalósíthatóság
  - Egyszerűség

A kriptográfiai verseny idővonala

- 1997.01.02: A NIST kihirdeti a kriptográfiai versenyt.
- 1997.04.15: A verseny kritériumainak megvitatása.
- 1997.09.12: A NIST bejelenti a felhívást az algoritmusok beküldésére.
- 1998.06.15: Beküldési határidő.
- 1998.08.20: A 15 jelölt bemutatása az Első AES Jelölt Konferencián.
- 1999.03.22–23: Második AES Jelölt Konferencia.
- 1999.08.09: A NIST bejelenti az 5 kiválasztott lehetséges algoritmust.
- 2000.04.13–14: A Harmadik AES Jelölt Konferencia.
- 2000.05.15: Javaslattételi szakasz vége.
- 2000.10.02: A NIST bejelenti a kiválasztott AES algoritmust.

A versenyre beküldött algoritmusok

Az első körben kieső algoritmusok

- CAST-256, *Carlisle Adams, Stafford Tavares, Howard Heys, Michael Wiener*
- CRYPTON, *Chae Hoon Lim*
- DEAL, *Lars Knudsen*
- DFC, *Henri Gilbert, Marc Girault, Philippe Hoogvorst, Fabrice Noilhan, Thomas Pornin, Guillaume Poupard, Jacques Stern, Serge Vaudenay*
- E2, *Masayuki Kanda, Shiho Moriai, Kazumaro Aoki, Hiroki Ueda, Miyako Ohkubo, Youichi Takashima, Kazuo Ohta, Tsutomu Matsumoto*
- FROG, *Dianelos Georgoudis, Damian Leroux, Billy Simón Chaves*
- HPC, *Richard Schroeppel*
- LOKI97, *Lawrence Brown, Josef Pieprzyk, Jennifer Seberry*
- MAGENTA, *Michael Jacobson Jr., Klaus Huber*
- SAFER+, *James Massey, Gurgen Khachatrian, Melsik Kuregian*

A versenyre beküldött algoritmusok

Az második körben kieső algoritmusok

- MARS, *Carolynn Burwick, Don Coppersmith, Edward D'Avignon, Rosario Gennaro, Shai Halevi, Charanjit Jutla, Stephen M. Matyas, Luke O'Connor, Mohammad Peyravian, David Safford, Nevenko Zunic*
- RC6, *Ron Rivest, Matt Robshaw, Ray Sidney, Yiqun Lisa Yin*
- Serpent, *Ross Anderson, Eli Biham, Lars Knudsen*
- Twofish, *Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, Niels Ferguson*

A győztes algoritmus

- **Rijndael**, *Vincent Rijmen, Joan Daemen*

A Rijndael tulajdonságai

- Kulcsméret: 128-tól 256 bitig, 32 bites lépésekben.
- Blokkméret: 128-tól 256 bitig, 32 bites lépésekben.
- A kulcs és a blokk mérete független.
- Az AES fix 128 bit méretű blokkokat és 128, 192, vagy 256 bites kulcsot határoz meg.
- A gyakorlatban az AES-128 (128 bites blokk és kulcs) és az AES-256 (128 bites blokk, 256 bites kulcs) használatos.

Elektronikus kódkönyv (ECB) mód

Működése

- A nyílt szöveget a blokkméretnek megfelelő darabokra vágjuk.
- Minden blokkot a titkosító kulccsal titkosítunk.
- Lényegében a blokkméretnek megfelelő betűket használó egybetű-helyettesítő kód. Hasonló módon támadható.

Passzív támadás

- A titkosított üzeneteket **lehallgatjuk**.
- Ugyanabból a nyílt szöveg blokkból mindig ugyanaz a titkosított szöveg blokk keletkezik.
- Sokféle adat tárolódik, vagy továbbítódik mezőkre osztva. Egy mező mérete könnyen lehet pontosan egész számú blokk.

Elektronikus kódkönyv (ECB) mód

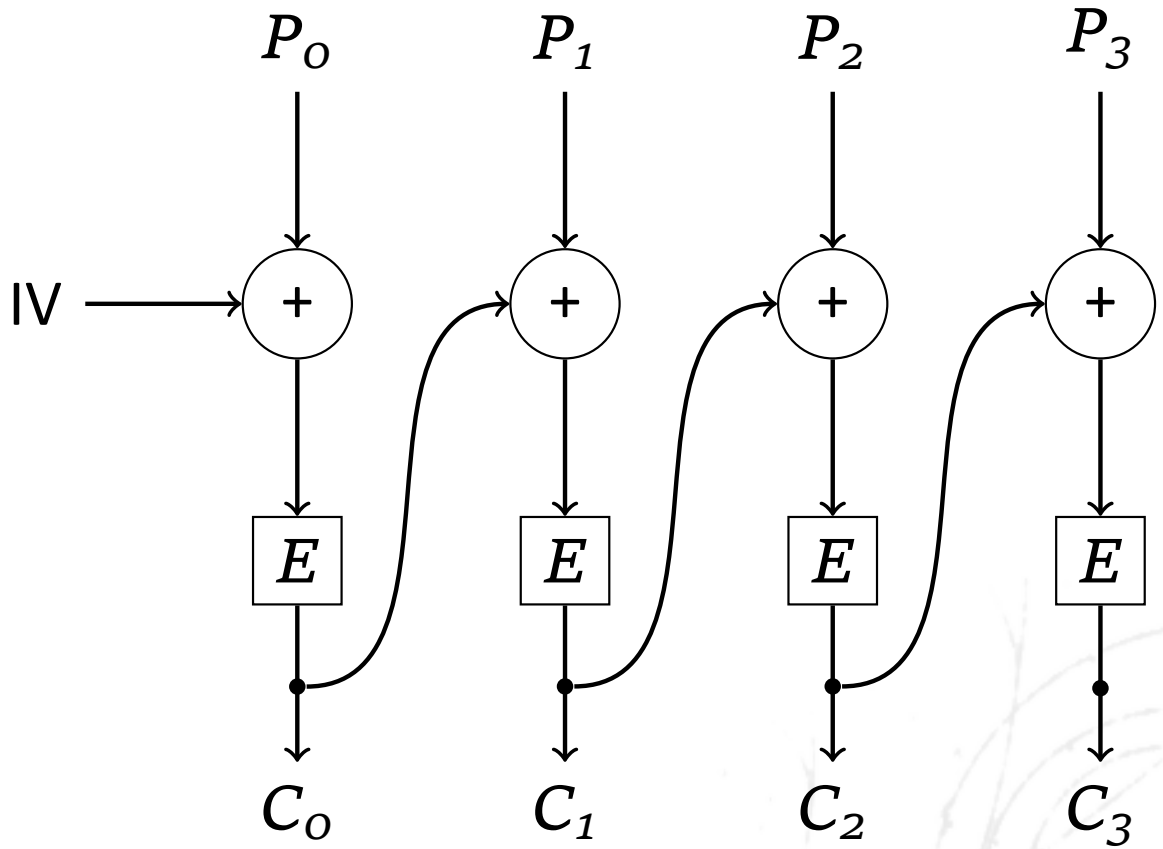
Aktív támadás

- A lehallgatott üzeneteket **visszajátszuk**, az üzenetek egy-egy részét régebbi lehallgatott darabokkal *kicseréljük*.
- Nem feltétlenül kell, hogy az üzenet pontos tartalmát ismerjük.
- Példa: Banki tranzakciós adatok mezőinek kicserélése

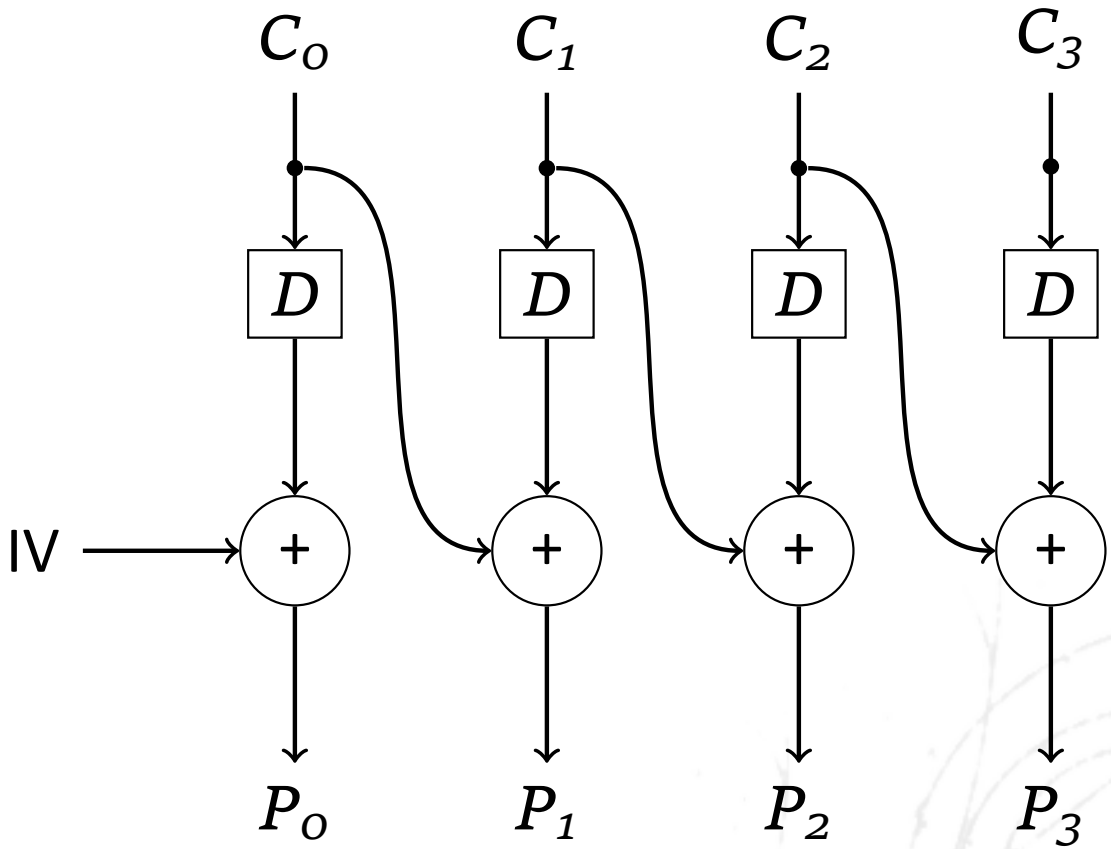
TItkosított blokkok láncolása (CBC)

- Minden titkosítandó blokkot kizáró vagy kapcsolatba hozunk az előző blokkhoz tartozó titkosított blokkal.
- A legelső blokk előtt nincs mivel XOR-olni, így ott egy inicializáló vektort használunk.
- Dekódoláskor ugyanezt tesszük, visszafelé.
- Az inicializáló vektort is továbbítani kell.

CBC mód, a titkosítás folyamata



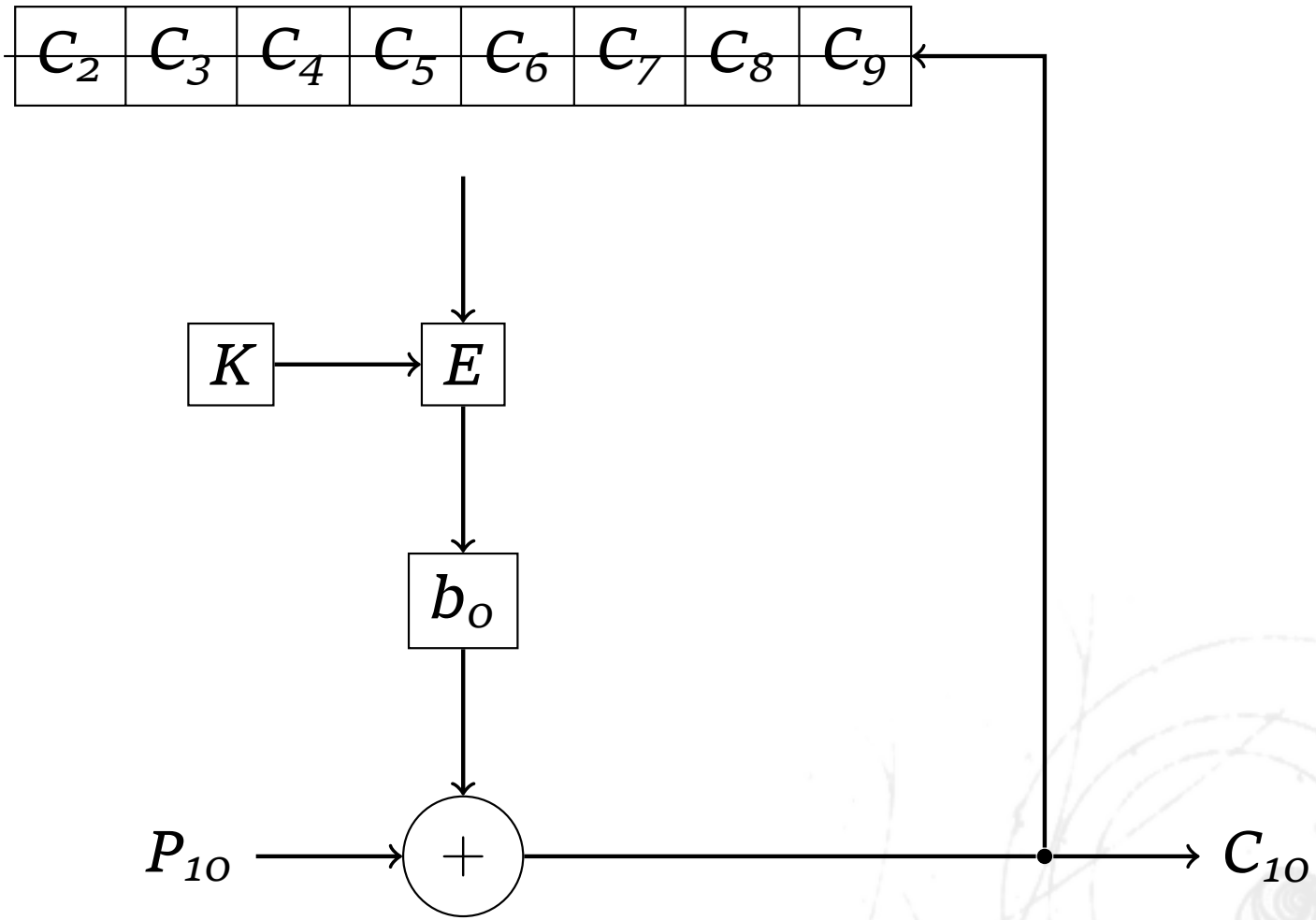
CBC mód, a dekódolás folyamata



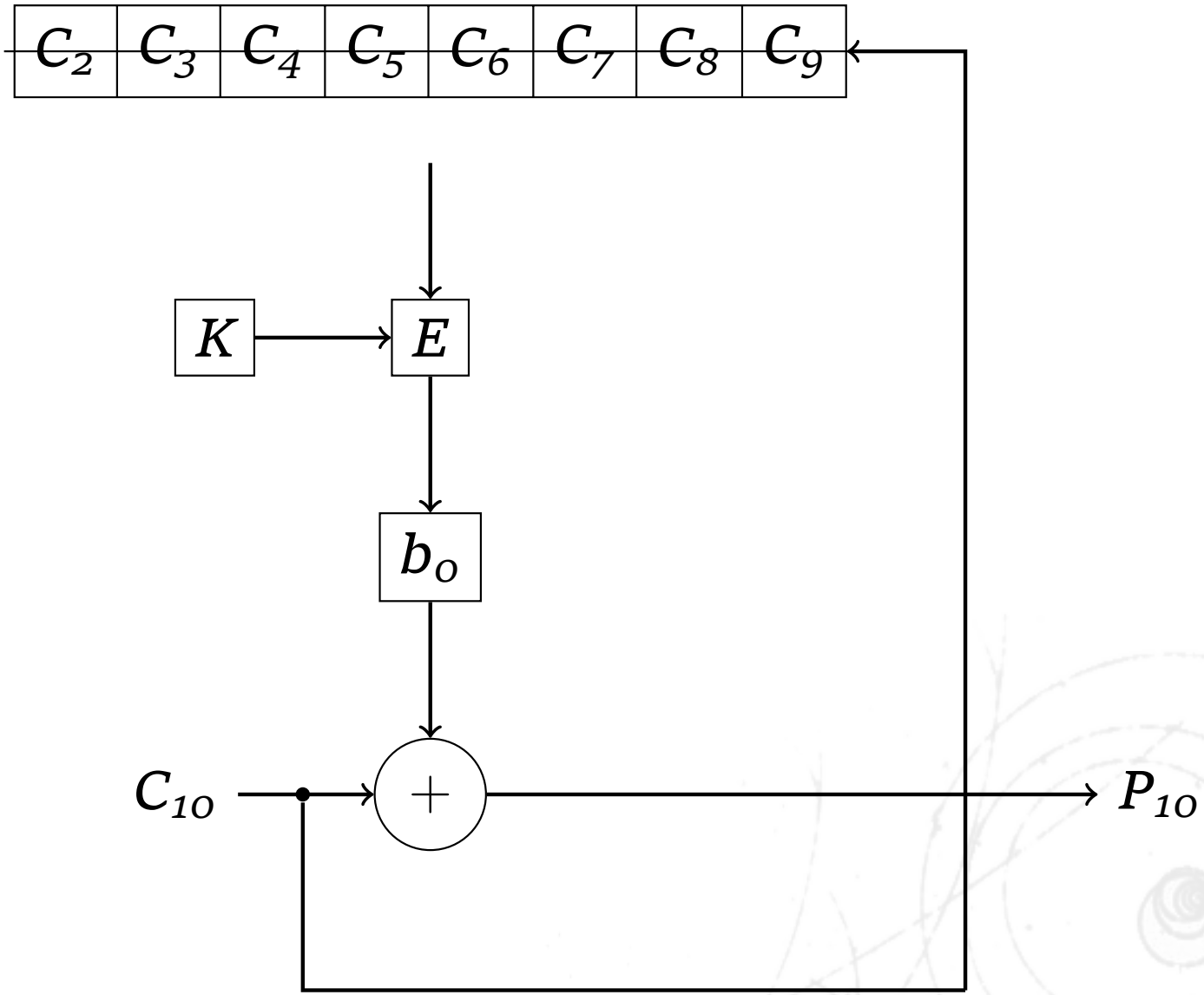
Kimenet visszacsatolása (OFB)

- A CBC mód hátránya, hogy mindig meg kell várni egy teljes 64 bites (DES, 3DES), vagy 128 bites (AES) blokkot ahhoz, hogy a következő blokkot titkosítani lehessen. Ezen segít az OFB mód.
- Nem közvetlenül a bemenő adatot titkosítjuk.
- Mind az adó, mind a vevő oldalon ugyanazt a számsort állítjuk elő, ezzel hozzuk bájtonként **kizáró vagy** kapcsolatba a titkosítandó adatot.
- A kimenő bájtokat egy **léptetőregiszterben** tároljuk, ennek aktuális tartalmát titkosítjuk.
- A kizáró vagy művelethez a titkosított adat (hosza egy egész blokk) bal szélső bájtjt használjuk fel.

Az OFB mód, a kódolás folyamata



Az OFB mód, a dekódolás folyamata



Folyamtitkosítási (CFB) mód

- Az OFB mód esetén egy 1 bájtos hiba után az összes adat használhatatlanná válik. Erre ad megoldást a CFB mód.
- Itt egy inicializáló vektort eltitkosítunk a kulccsal. Az így kapott blokkot újra eltitkosítjuk. Stb. Így egy kulcsfolyamot kapunk.
- A kulcsfolyamot mind az adó, mind a vevő előállítja.
- A tulajdonképpeni titkosítás itt is kizáró vagy művelettel történik a nyílt szöveg és a kulcsfolyam között.

Számláló (CTR) mód

- A CFB mód közvetlen hozzáférésre nem alkalmas (nagy fájl, teljes lemez titkosítása).
- Ezt a problémát oldja meg a számláló mód.
- Az n-edik blokk titkosítása a következő módon zajlik:
  - Az inicializáló vektorhoz hozzáadunk n-et
  - Az $IV + n$ értéket titkosítjuk a kulccsal
  - Az így kapott értékkel hozzuk kizáró vagy kapcsolatba a nyílt szöveg n-edik blokkját.
- Mind az adó, mind a vevő oldalon ugyanazt az inicializáló vektort használjuk.