Kriptográfia

Negyedik előadás
A DES

Németh L. Zoltán
SZTE, Számítástudomány Alapjai Tanszék
2008 ősz

Modern Blokktitkosítók

- az eddig módszerek mind feltörhetőek
- akkor hogyan titkosítsunk?
- modern (gépi) módszerek következnek
- ma is a szimmetrikus blokktitkosítók a legelterjedtebbek (gyorsak, sokat vizsgáltak)
- titkosságot / hitelességet biztosítanak
- a **DES**-re (Data Encryption Standard, 1977) fogunk koncentrálni
- hogy bemutassuk a blokktitkosítók általános tervezési elveit

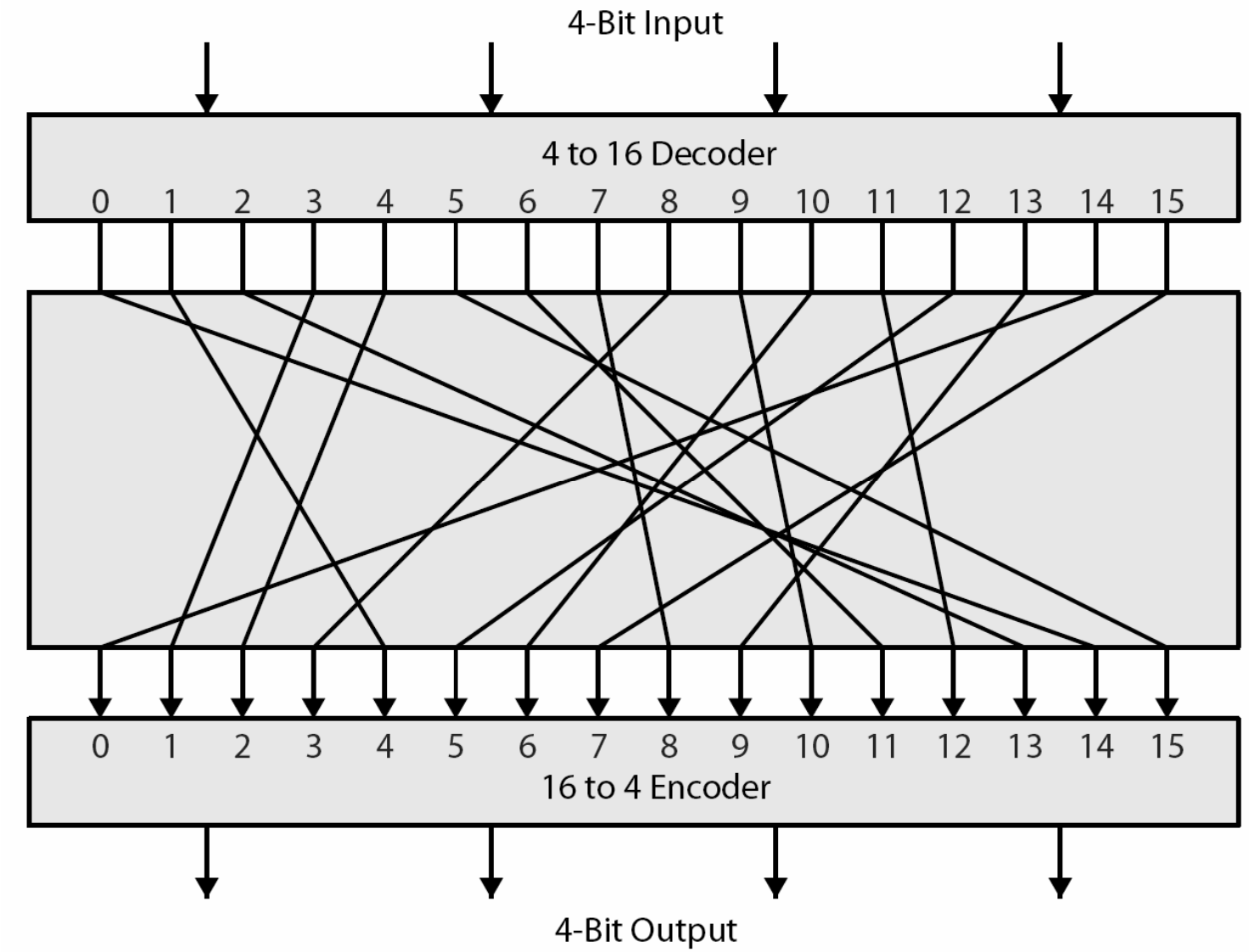
A blokkok mérete

- a blokk titkosítók az üzenetet fix hosszúságú egységenként titkosítják/fejtik meg
- tekinthetjük őket hatalmas ábécék feletti helyettesítésnek (64-bites vagy még több)
- elég sok bitre szükség van, hogy a helyettesítés feltörésére alkalmazott gyakoriságanalízisnek ne legyen esélye
- a DES blokkmérete 64 bit
- az AES blokkmérete 128 bit

Block vagy folyam titkosítók ?

- a folyamtitkosítók (stream ciphers) az üzenetet bitenként vagy bájtonként titkosítják / fejtik meg
- ekkor nem kell egy blokknyi adatot összevárni a titkosítás megkezdéséhez
- számos mai titkosító blokktitkosító
- szélesebb körben alkalmazottak, mint a folyamtitkosítók
- blokkonként lehet velük adatfolyamot is továbbítani
- majd később nézzük a működési módjaikat

Az ideális blokktitkosító



A példa által definiált helyettesítés

Plaintext	Ciphertext
0000	1110
0001	0100
0010	1101
0011	0001
0100	0010
0101	1111
0110	1011
0111	1000
1000	0011
1001	1010
1010	0110
1011	1100
1100	0101
1101	1001
1110	0000
1111	0111

Ciphertext	Plaintext
0000	1110
0001	0011
0010	0100
0011	1000
0100	0001
0101	1100
0110	1010
0111	1111
1000	0111
1001	1101
1010	1001
1011	0110
1100	1011
1101	0010
1110	0000
1111	0101

Blokk titkosítók alapelvei

- a blokktitksosítók hatalmas ábécék feletti helyettesítések
- de szükség van rá, hogy a helyettesítés injektív legyen, sőt effektíven kiszámítható
- általános esetben 64 biten $2^{64}!$ darab helyettesítés definiálható
- egy helyettesítés leírásához egy 2^{64} soros 64-bites értékeket tartalmazó táblázat kell, ami $64 \cdot 2^{64} = 2^{70} \approx 10^{21}$ bites kulcsméretet jelent.
- hogyan készíthető akkor biztonságos invertálható helyettesítés kisebb komponensekből?
- a legtöbb szimmetrikus blokktitkopsító ún.

  Feistel titkosító stuktúrán alapszik

Helyettesítő-keverő titkosítók (Substitution-Permutation Ciphers)

- Claude Shannon vezette be a helyettesítő-keverő hálózatokat (S-P networks) 1949-ben
- a modern blokktitkosítók rajtuk alapulnak
- az S-P hálózatok pedig a két korábbi alap kriptográfiai műveletből épülnek fel:
  - *helyettesítés (S-doboz) / substitution* (S-box) /
  - *keverés (P-doboz) / permutation* (P-box) /
- ezek biztosítják a titkos szövegnek mind a nyílt szövegtől, mind a kulcstól való minél komplexebb, de könnyen megadható függését

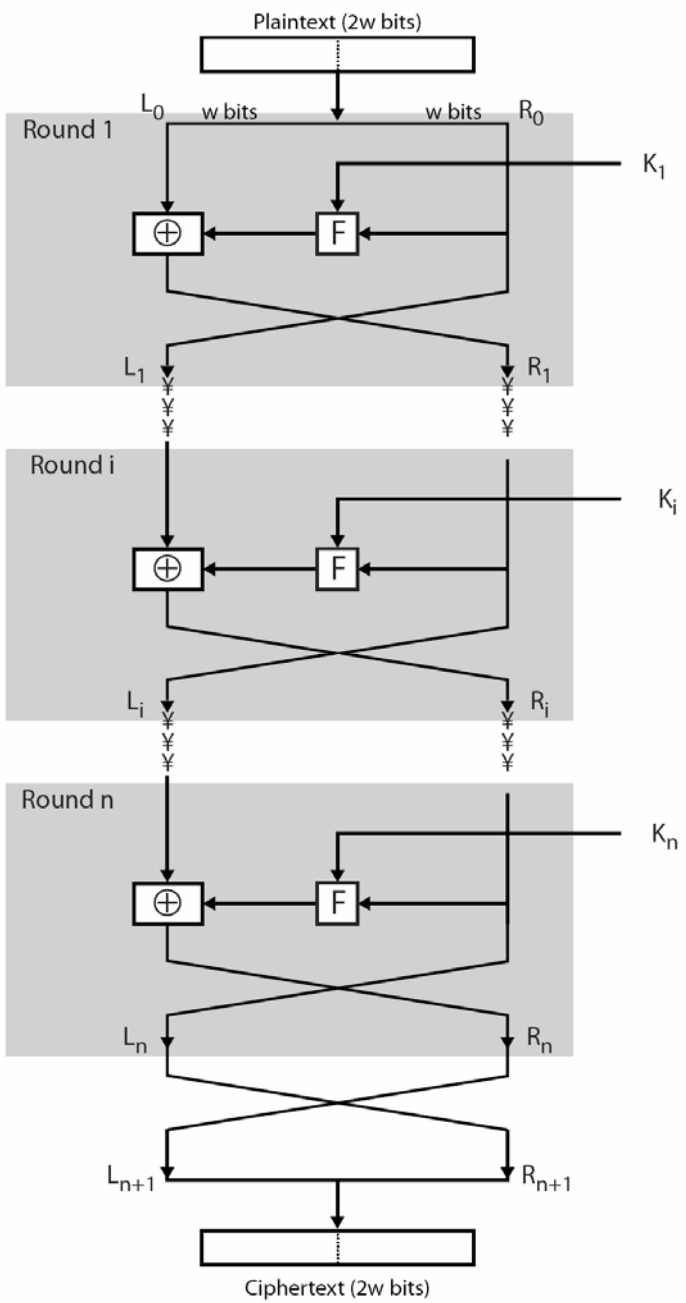
Diffúzió és konfúzió (Confusion and Diffusion)

- a titkosítónak a nyílt szöveg statisztikai jellemzőit a felismerhetetlenségig el kell rejtenie a titkosított szövegben
- mint például a OTP, és az ideális blokktitkosító
- a gyakorlatban erre Shannon az S és P dobzok kombinálását javasolta, a két cél:
- **diffúzió (szétterjesztés)** – szétoszlatja a nyílt szöveg statisztikai struktúráit egy terjedelmes méretű titkos szöveg részekben
- **konfúzió (összekeverés)** – a titkos szöveg és a kulcs kapcsolatát minél komplexebbé teszi

A Feistel titkosító struktúra

- Horst Feistel ötlete a 70-es évek elején
  - Shannon (S-P hálózatos) produkciós titkosítós ötletéből
  - könnyen invertálható titkosítók általános terve
  - más körfüggvény használatával más tikosítót kapunk
- a bemenő blokkot két félre osztja (pl. 32-32 bit)
- a titkosítás több körben (round) zajlik, melyekben
  - az adatok bal felén helyettesítést hajt végre (XOR-ol)
  - melyhez a körfüggvényt (round function) használja, ami
  - az adatok jobb felének és a körkulcsnak a függvénye
  - végül a bal és jobb feleket felcseréli (ez permutáció)

A Feistel Struktúra vázlata

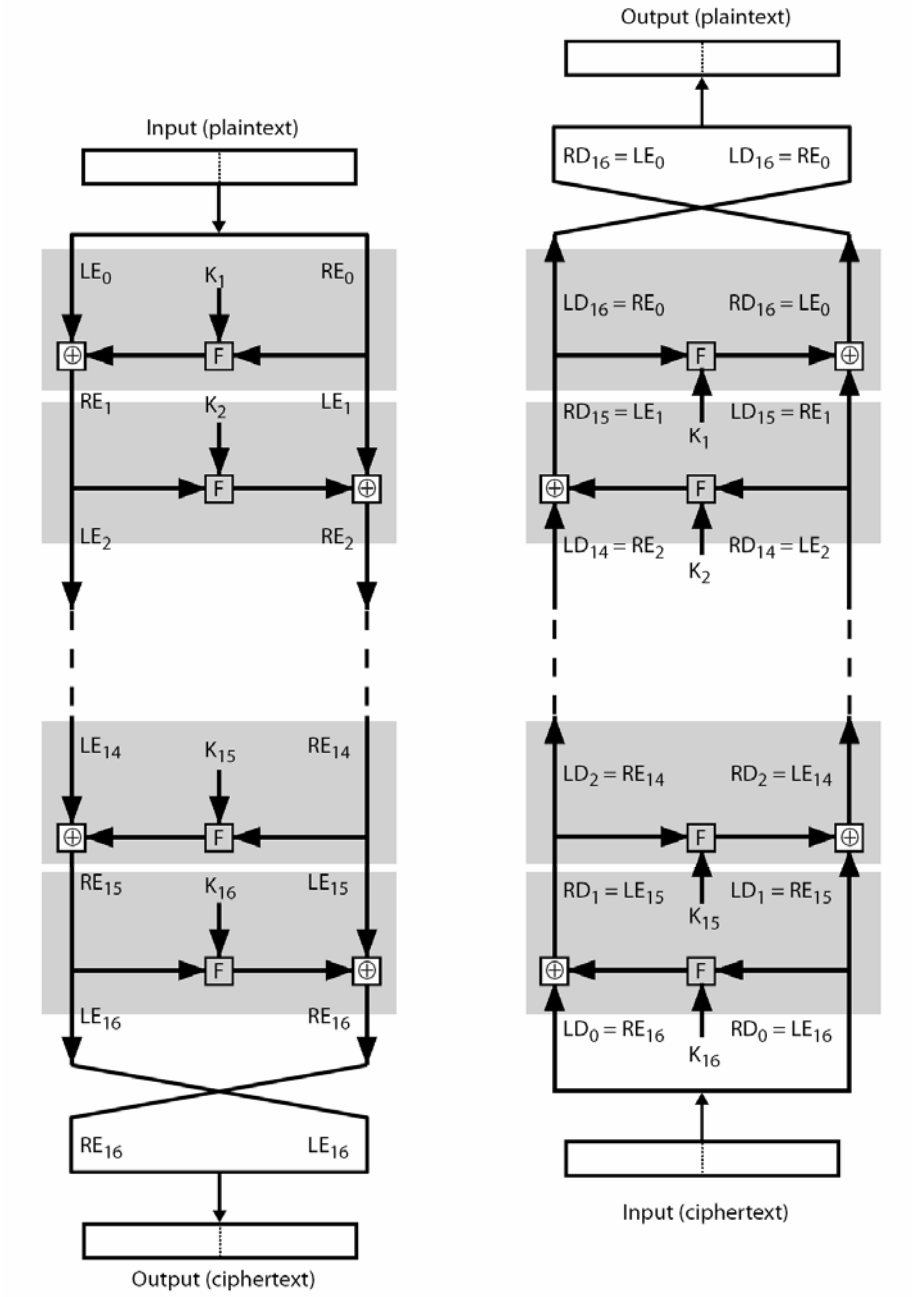


Miért jó ez?

Ez a struktúra azért (is) kedvező, mert

- míg tipikusan egy kör csak gyenge (egyszerű) titkosítást ad,
- a körök egymás utáni alkalmazása jelentősen növeli az erősséget
- a megfejtés algoritmusa majdnem ugyanaz,
- csak a körkulcsokat kell fordított sorrendben alkalmazni (ld. a következő ábrát)
- ezért került az utolsó kör végére még egy csere

Titkosítás (bal) és megfejtése (jobb)



Feistel struktúrák tervetésének elemei

- blokkméret
- kulcsméret
- körök száma (pl. DES-nél 16 kör van)
- körkulcs generálás algoritmusa
- körfüggvény (további S és P dobozok)

Figyelembe kell venni:

- gyors szoftveres/hardveres megvalósíthatóságot
- könnyű legyen analizálni

Data Encryption Standard (DES) (Adat titkosítási szabvány)

- a világon ma is a legelterjedtebb blokktitkosító
- 1977-ben vezették be az NBS (National Bureau of Standards, ma NIST)
  - FIPS 46 szabvány: http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf
- 64-bites adatblokk, 56-bites kulcssal
- biztonsága sok vitát váltott ki
- bár (nyilvános) ismereteink szerint csak brute force módon törhető
- ma már új általános szabvány is van az AES (128, 196 vagy 256 bites kulccsal)

A DES története I

- az IBM kifejleszti a Lucifer titkosítót
  - Feistel vezette csoport,
  - 60-as évek végétől 1971-ig
  - 64-bites blokkonklént 128-bites kulccsal
- majd újra tervezték:
  - kereskedelmi célokra
  - NSA (National Security Agency = az USA Nemzetbiztonsági Hivatala) és más szakértőkkel közösen
  - férjen el egyetlen chipen -> csak 56 bites kulcs

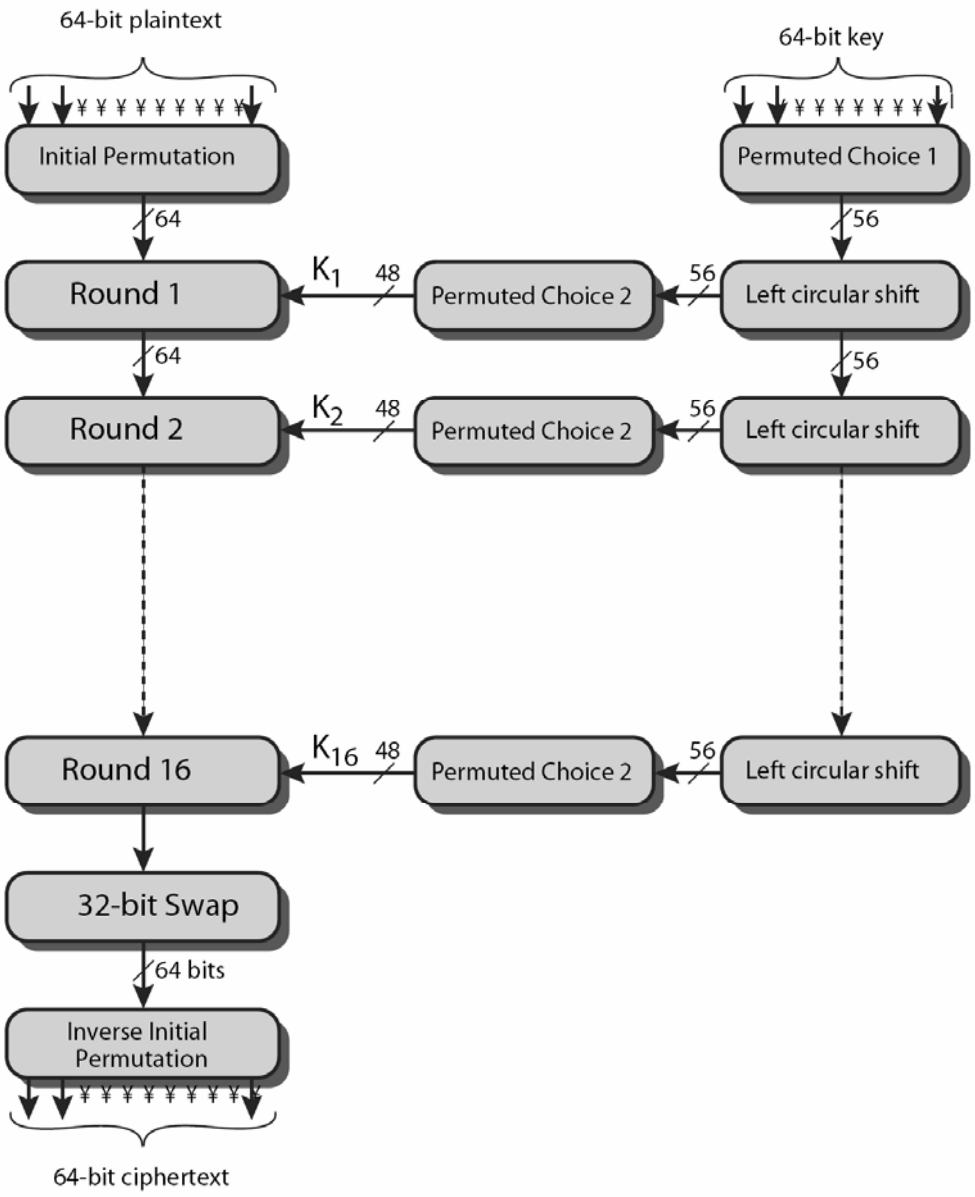
A DES története II

- 1973-ban NBS pályázatot hirdetett az USA-ban nemzeti titkosító algoritmusra
- az IBM a Lucifer módosított változatát nevezte
- melyet el is fogadtak, mint Data Encryption Standard-ot, azaz adattitkosítási szabványt 1977-ben

Viták a DES tervezéséről

- annak ellenére, hogy a DES nyilvános szabvány
- vitákra adott okot tervezésében
  - a kulcs 56-bitre választása (a Luciferé 128-bit)
  - a részletes tervezési kritériumokat (pl. az S dobozokét) nem hozták nyilvánosságra
- a későbbi analízis mégis azt mutatja, hogy jól tervezett, bevált rendszer
- széles körben alkalmazzák ma is
  - különösen pénzügyi alkalmazásokban
  - használata ma is szabványos meglevő rendszerekben
  - új alkalmazásokban, persze már modernebb titkosítást (pl. AES-t, vagy 3DES-t) kell bevezetni

A DES titkosításának áttekintése



A kezdeti keverés (Initial Permutation IP)

- ez a legelső lépés a 64 bites adatblokkon
- IP átrendezi a biteket sorrendjét, közben
- a páros sorszámú bitek a bal félbe (LH)
- a páratlanok a jobb félbe (RH) kerülnek
- meglehetősen szabályos strukturát mutat (felülről lefele, majd jobbról-balra)
- például:

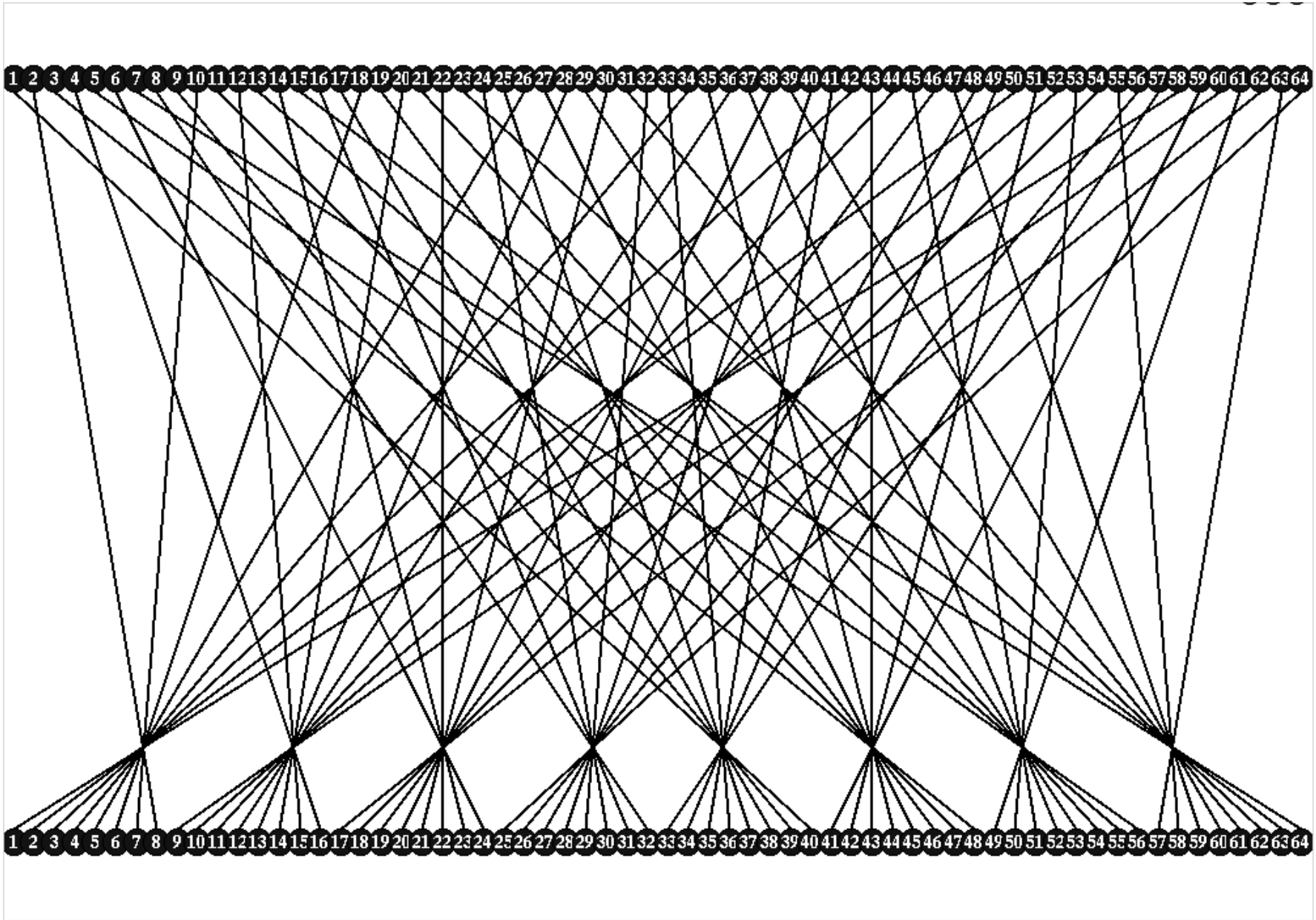
```
IP(675a6967 5e5a6b5a) = (ffb2194d 004df6fb)
```

A kezdeti keverés táblázata

IP

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

A kezdeti keverés grafikusan

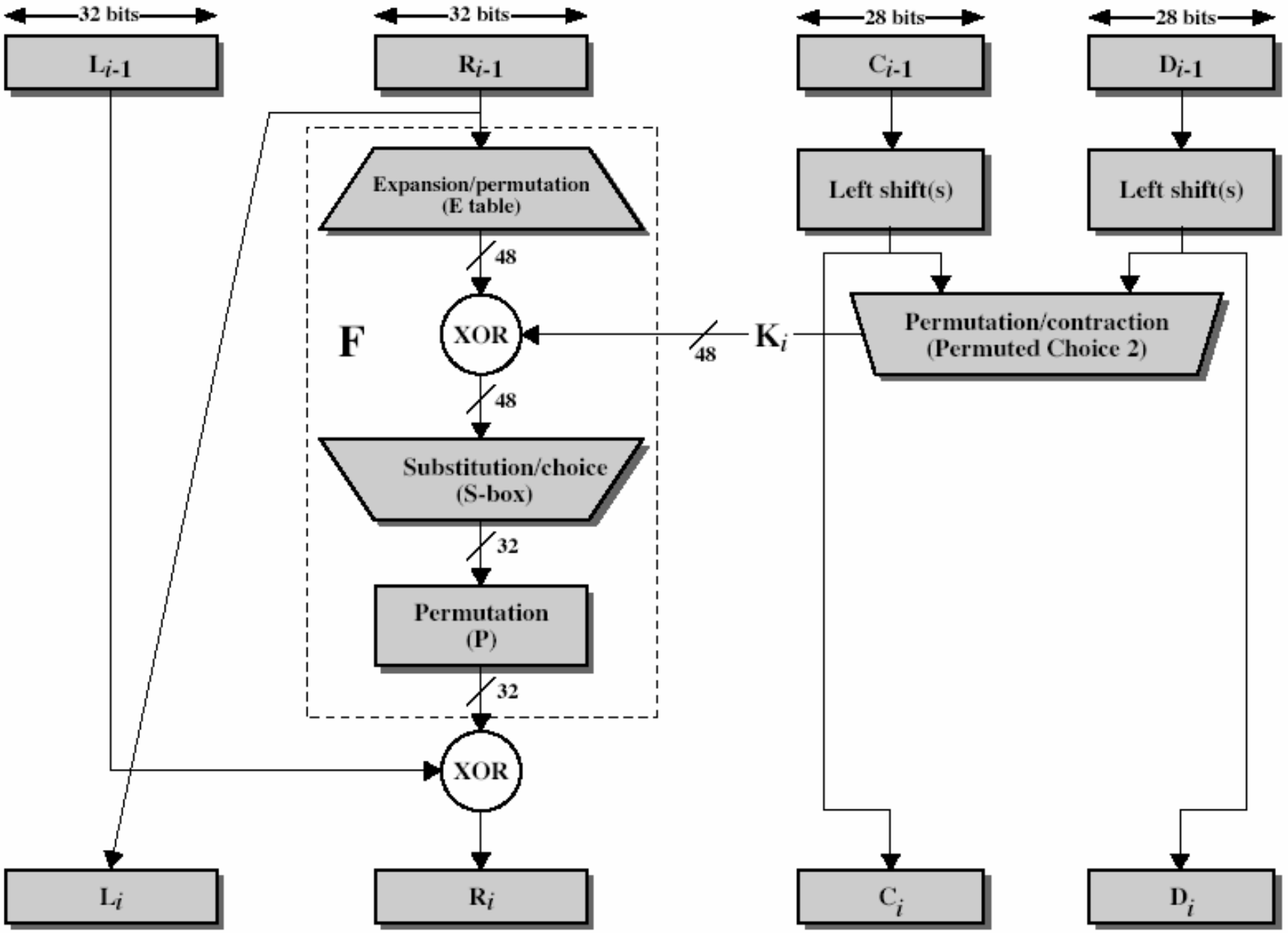


A kezdeti keverés inverze

IP^{-1}

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

A DES egy köre



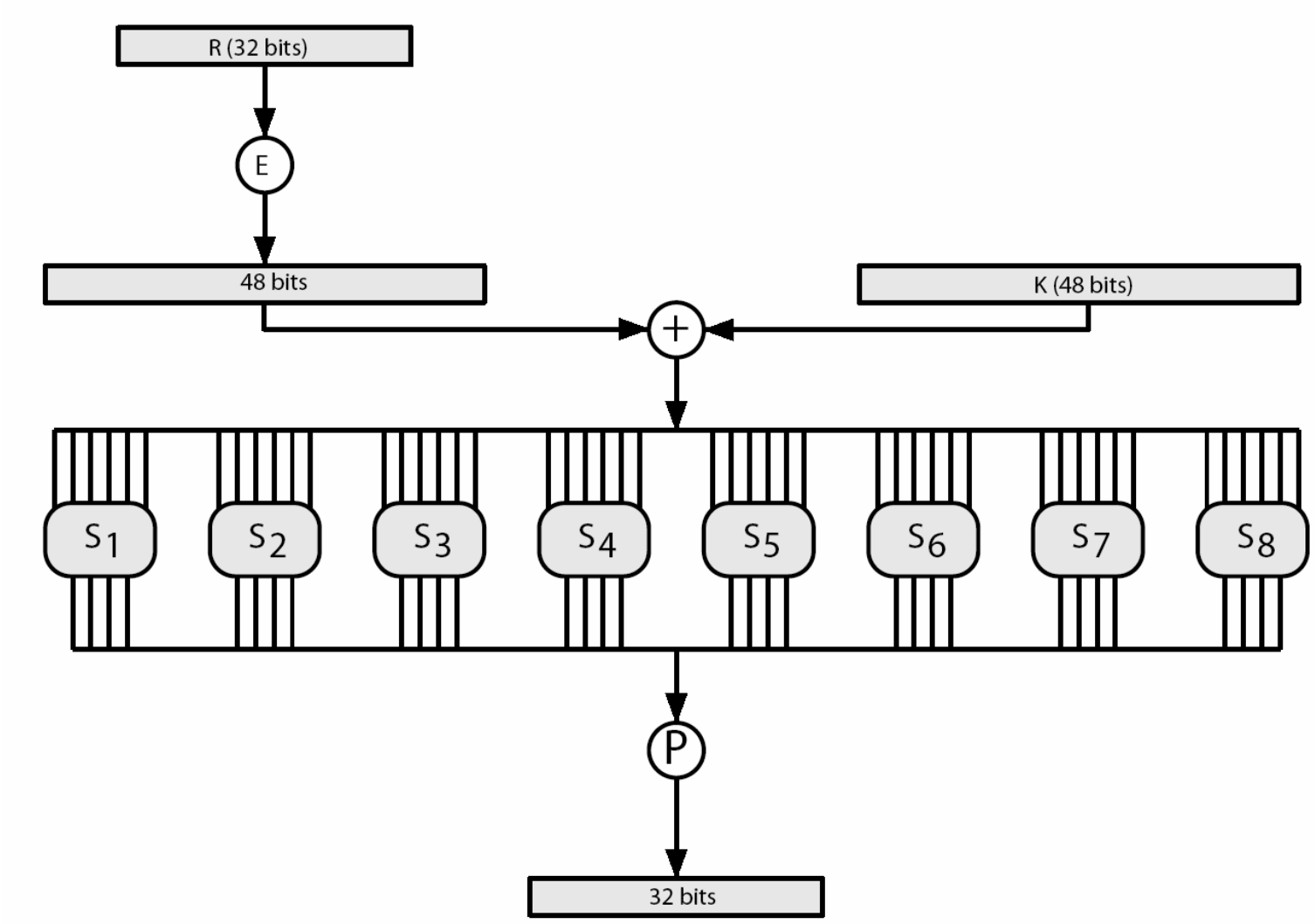
A DES köreinek felépítése

- a két fél 32-bites blokk az *i*-dik körben: L_i és R_i
- ahogy minden Feistel struktúrában itt is:

  $L_i = R_{i-1}$

  $R_i = L_{i-1} \oplus \mathrm{F}(R_{i-1}, K_i)$
- **F körfüggvény** így egy 32-bites R blokkfelet és egy 48-bites körkulcsot kap és
  1. kiterjeszti R-et 48-bitre az **E**-segítségével (expansion)
  2. **XOR** műveletet végez vele a körkulccsal
  3. a **8 darab** egyenként 6-bemeneti és 4 kimeneti bites **S-dobozzal** a 48=8·6 bitet 8·4 = 32 bittel helyettesíti
  4. végül a **P pemutációval** összekeveri a 32 bitet

A DES körfüggvénye



Az S-dobozok

- a 8 S-doboz mindegyike 6-ból 4 bitet készít
- igazából minden S-doboz 4 darab „4-ből 4 bites” kis dobozból áll
  - a külső bitek az 1. és a 6. (**sor bitek**) kijelölnek egyet a 4 sorból (azaz a 4 kisdobozból)
  - a belső bitek 2-5. (**oszlop bitek**) e sor szerint helyettesítődnek
  - így egy S-doboz egy 4 (soros) x 16 (oszlopos) táblázat ír le, melynek elemei 4 bites értékek (0-15 számok)
- pl hexadecimális jelöléssel :
  `S(`**`18 09`**` 12 3d 11 17 38 39) = `**`5f`**`d25e03`

  S1(011000)=**5**, mert S1 00 sor 1100 oszlopában 5 van

  S2(001001)=**f** mert S2 01 sor 0100 oszlopában **15=f** van

Az S1 doboz

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

Az S2 doboz

15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S dobozok tervezése

- az S-dobozok a DES egyedüli nemlineáris komponensei
- tervezésük különös figyelemmel történt
- ezek tervezési részleteit nem hozták nyilvánosságra
- pl. a differenciális kriptoanalízis ellen védelmet nyújtanak, amely módszer csak 90-ben került nyilvánosságra
- általában a biztonságos S-dobozok tervezése a kriptográfia egy külön fejezete

A DES S dobozainak tulajdonságai

- nem lineárisak
- soraikban a 0..15 számok mindegyike pontosan egyszer szerepel
- a bemenetet 1 bittel változtatva a kimenet legalább 2 bitben megváltozik
- $S(x)$ és $S(x \oplus 001100)$ legalább 2 biten eltér
- $S(x) \neq S(x \oplus 11ab00)$ minden $a,b \in \{0,1\}$ -re

DES kulcsszervezés (Key Schedule)

- hogyan kapjuk a körkulcsot az eredetiből?
  - a szabványos kulcs 64 bit, de ebből a kezdeti (kulcs) keverés PC1 (Permuted Choice 1) csak 56 bitet választ ki, ezt két 28 bites félre osztjuk
  - a 16 db körkulcs generálás lépései:
    - forgassuk mindkét felet külön-külön balra 1 vagy 2 bittel (a forgatás mértékét egy táblázat adja meg)
    - válasszunk ki 24-bitet mindkét félből és permutáljuk őket PC2 (Permuted Choice 2) segítségével
- vegyük észre a könnyű hw/sw megvalósíthatóságot

A kör kulcsgeneráláshoz szükséges forgatások táblázata

Round number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bits rotated	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Összesen 28 forgatás,
éppen körülfordul a két fél körkulcs

Crypt Tool DES animáció

A DES megfejtése

- a megfejtéskor fordítva kell alkalmazni a titkosítás lépéseit az adatokon
- mivel a kezdeti és végső keverés egymás inverze
- a többi pedig Feistel-struktúra
- ezért elég a körkulcsokat (jobbra forgatással) fordítva K_{16} … K_1 sorrendben generálva megismételni a titkosítást
  - IP visszarendezi a végső keverés FP hatását
  - az 1. kör az K_{16} visszafejti a 16. kör titkosítását
  - ….
  - az 16. kör az K_1 visszafejti a 1. kör titkosítását
  - végül a végső keverés (FP) hatástalanítja a kezdeti keverést (IP)
  - így visszakapjuk a nyílt szöveget

A lavina hatás

- az egyik igen kívánatos tulajdonsága bármely titkosításnak, hogy
- a nyílt szöveg vagy a kulcs kisméretű változtatása is, jelentősen változtassa meg a titkosított szöveget.
- 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
- A DES erős lavinahatással rendelkezik, mint az a következő táblázat mutatja

A DES lavinahatásának szemléltetése

a tömör négyzetek a megválzozott bitek

Lavina hatás a DES-ben

(a) Change in Plaintext		(b) Change in Key	
Round	Number of bits that differ	Round	Number of bits that differ
0	1	0	0
1	6	1	2
2	21	2	14
3	35	3	28
4	39	4	32
5	34	5	30
6	32	6	32
7	31	7	35
8	29	8	34
9	42	9	40
10	44	10	38
11	32	11	31
12	30	12	33
13	30	13	28
14	26	14	26
15	29	15	34
16	34	16	35

Támadások a DES ellen I
Brute-force

- **kis kulcsméret** a DES leggyengébb (ismert) pontja
- 56-bites kulcsból $2^{56} = 7.2 \times 10^{16}$ darab van
- ezzel látszólag ellenáll a teljes kipróbálásnak
- ám komoly erőfeszítéssel a brute-force módon feltörhető
  - 1997 Internet (ditributed.net) 96 nap
  - 1998 speciális hardwer (Deep Crark gép /EFF/) 56 óra
  - 1999 (együtt: Deep Crack gép + ≈100000 PC) 22 óra!
- persze ehhez képesnek kell lenni a nyílt szöveg felismerésére is
- és ma már mindenképpen a DES-nek alternatíváit kell keresnünk, pl. AES, 3DES

A Deep Crack gép

- Speciális hardwer a DES brute-force módú feltörésére
- Már 1977-ben Diffie és Hellman tervezett papíron egy ilyen gépet, ami akkor 20 millió $-ba került volna
- 1998-ban az Electronic Frontier Foundation épített is egy ilyet kevesebb mint 250.000 $-ból
- Ez a Deep Crack Machine
  - http://en.wikipedia.org/wiki/Deep_Crack
- a tervet nyilvánosságra is hozták
- ma már pár új autó áráért készíthető olyan berendezés, ami a DES-t 1 napon belül feltöri, avagy olcsóbban:
- 2006: 8980 Euro, 9 nap, COPACOBANA (Cost-Optimized Parallel COde Breaker)
  - http://www.copacobana.org/

Támadások a DES ellen II
egyéb támadások

- elég sok ismert, de a gyakorlatban kivitelezhetetlenek pl. rengetek ismert vagy választott nyíltszöveg kell hozzájuk, pl.
  - differenciális kriptoanalízis
    - 2^{47} DES művelet, 2^{47} választott nyílt szöveggel
  - lineáris kriptoanalízis
    - 2^{39} - 2^{41} DES művelet, 2^{43} ismert nyílt szöveggel
- időméréses támadás (timing attack)
  - konkrét implementációt elemez, az alapján, hogy a titkosítás ideje függ a bemenettől és a használt kulcstól
- ha a kulcs néhány bitje már meghatározott a maradék teljes kipróbálással kereshető

A kétszeres DES és feltörése

- a DES igazán gyenge pontja az 56 bites kulcsméret
- ezt megpróbálhatjuk orvosolni, úgy, hogy a DES titkosítást egymás után **többször alkalmazzuk**, különböző kulcsokkal
- **két egymásutáni DES-t végezni kevés**, mert
- **középen találkozó (meet in the middle)** támadással feltörhető:
- néhány ismert nyílt szöveg-titkosszöveg pár birtokában
  - a nyíltszöveget minden DES kulccsal titkosítva
  - a párját, minden DES kulccsal megfejtve
  - táblázatot vezetve az értékekről
  - az egyezéseket találhatunk
  - melyeket a többi párra ellenőrizve
- 2^{57} lépésben feltörhetjük a 112 bites kulcsot
  - táblázat a mérete az idő rovására csökkenthető !

Háromszoros DES (Triple DES vagy TDES)

- Így marad a DES 3-szor egymás után történő alkalmazása
- $E_{K1}(E_{K2}(E_{K3}(x)))$ titkosítás helyett, azonban
- szerencsésebb az $\mathbf{E_{K1}(D_{K2}(E_{K3}(x)))}$ forma
  - mert ekkor K1=K2=K3 alkalmazásával a módszer kompatibilis marad a **DES**-sel
  - ha mindhárom kulcs különböző akkor **3TDES**
  - ha K1=K3≠K2, akkor **2TDES**
- mivel a titkosítás és megfejtés algoritmusa csak a kulcs kezelésben tér el ez nem befolyásolja a biztonságot

Háromszoros DES (Triple DES vagy TDES)

- így a 3TDES kulcsa **K1K2K3 168 bit** paritásbitekkel 192 bit

  $E_{K1K2K3}(x) = E_{K1}(D_{K2}(E_{K3}(x)))$

  $D_{K1K2K3}(x) = D_{K3}(E_{K2}(D_{K1}(x)))$

- így a 2TDES kulcsmérete **K1K2 112 bit** paritásbitekkel 128 bit

  $E_{K1K2}(x) = E_{K1}(D_{K2}(E_{K1}(x)))$

  $D_{K1K2}(x) = D_{K1}(E_{K2}(D_{K1}(x)))$

A TDES biztonsága

- a középen találkozó feltörés miatt a **3TDES** effektív biztonsága amúgy is csak **112 bit** a
- **2TDES**-t biztonságát ismert támadások miatt pedig csak **80 bit**-re teszik
- ez ma még elegendőnek tűnik
- de sajnos a sebessége a DES-énak 3-szorosa
- az az AES pedig kb. 6-szor gyorsabb, nagyobb blokkmérettel és persze jóval nagyobb biztonsági ráhagyással rendelkezik
- így a TDES is lassan hátérbe szorul
- kivéve talán az elektronikus fizetés területén, ahol számos szabványba beépítették (elsősorban hardveres megvalósításokban)

Felhasznált irodalom

- Virrasztó Tamás: Titkosítás és adatrejtés: Biztonságos kommunikáció és algoritmikus adatvédelem, NetAcademia Kft., Budapest, 2004. Online elérhető: http://www.netacademia.net/book.aspx?id=1# (3.1. fejezet)
- William Stallings: Cryptography and Network Security, 4th Edition, Prentice Hall, 2006. (Chapter 3)
- Lawrie Brown előadás fóliái (Chapter 3)