

Kriptográfia

Ötödik előadás Az AES

Dr. Németh L. Zoltán

SZTE, Számítástudomány Alapjai Tanszék

2008 ősz

Az AES pályázat

- a DES szabványt le kellett váltani, mert
 - az 56 bites kulcs brute-force módon való feltörését több kísérlet is igazolta (distributed.net, Deep Crack)
 - kriptóanalízisen alapuló támadások is ismertek rá (igaz csak elméletiek)
- a TDES helyettesítheti, de lassú, és a blokkméret továbbra is kicsi (64 bit)
- az USA-ban a NIST újabb pályázatot hirdetett 1997-ben
AES = Advanced Encryption Standard-re
(Továbbfejlesztett titkosítási szabványra)

Az AES pályázat követelményei

- titkos kulcsú szimmetrikus blokktitkosító
- 128-bites blokk, 128/192/256-bites kulcs
- erősebb és gyorsabb legyen a TDES-nél
- hatékony megvalósíthatóság
 - számítási teljesítmény
 - kód és adatmemória szükséglet
 - előkészítő számítások (pl. kulcsszervezés)
- flexibilitás az egyes platformok és későbbi fejlesztések tekintetében
- aktív működés 20-30 évre (+ archív használat)

A nevezéshez szükséges dokumentumok

1. az algoritmus specifikációjának komplett leírása
2. a teljesítményekre vonatkozó számítások, becslések, mérések
3. teszt-vektorok (nyílt–titkosított szövegpárok)
4. analízis az ismert támadási módszerekkel szembeni helytállásról (pl. lineáris és differenciális kriptóanalízis)
5. az algoritmus lehetőségei és korlátai
6. referencia megvalósítás ANSI C-ben
7. optimalizálási lehetőségek C illetve JAVA megvalósításban

AES értékelési szempontok

- kezdeti kritériumok (az első két fordulóban):
 - biztonság – a gyakorlati kriptóanalízissel szemben
 - idő/tár igény – számítási hatékonyság szempontjából
 - az algoritmus és az implementáció tulajdonságai
- végső kritériumok (az 5 döntős résztvevőnek)
 - általános biztonság (az analízis nyilvános volt!)
 - a softveres és hardveres megvalósítás könnyűsége (korlátozott RAM/ROM memória esetén is)
 - az implementáció támadhatósága
(pl. az idő vagy áram felhasználás mérése alapján)
 - flexibilitás (titkosítás/megfejtés, kulcsszervezés/csere, paraméterek, platformok, utasítás szintű párh.-síthatóság)

Az AES pályázat eredménye

- az értékelés három fordulóban zajlott
- 1998 jún: az első fordulóban 15 jelelőltet fogadtak el
- 1999 aug: a második fordulóban ezt 5-re szűkítették
- 2000 okt: a **Rijndael** lett a győztes (ejtsd:
www.iaik.tu-graz.ac.at/research/krypto/AES/wav/rijndael_pronunciation.wav)
- 2001 nov: **FIPS 197** szabványként publikálták **AES** néven
- 2003 jún: az NSA minősített adatok védelmére is engedélyezi: Secret, Top Secret
(utóbbira csak 192/256 bites kulccsal)
- ez az első ilyen mindenki számára elérhető titkosítás

AES szűkített lista

- Az öt döntős résztvevő:
 - MARS (IBM) - complex, gyors, nagy bizt. ráhagyás
 - RC6 (USA) - n. egyszerű, n. gyors, kis bizt. ráhagyás
 - Rijndael (Belgium) - világos, gyors, jó bizt. ráhagyás
 - Serpent (Euro) - világos, lassú, n. nagy bizt. ráhagyás
 - Twofish (USA) - complex, n. gyors, nagy bizt. ráhagyás
- ezután jött az újabb szempontok analízise

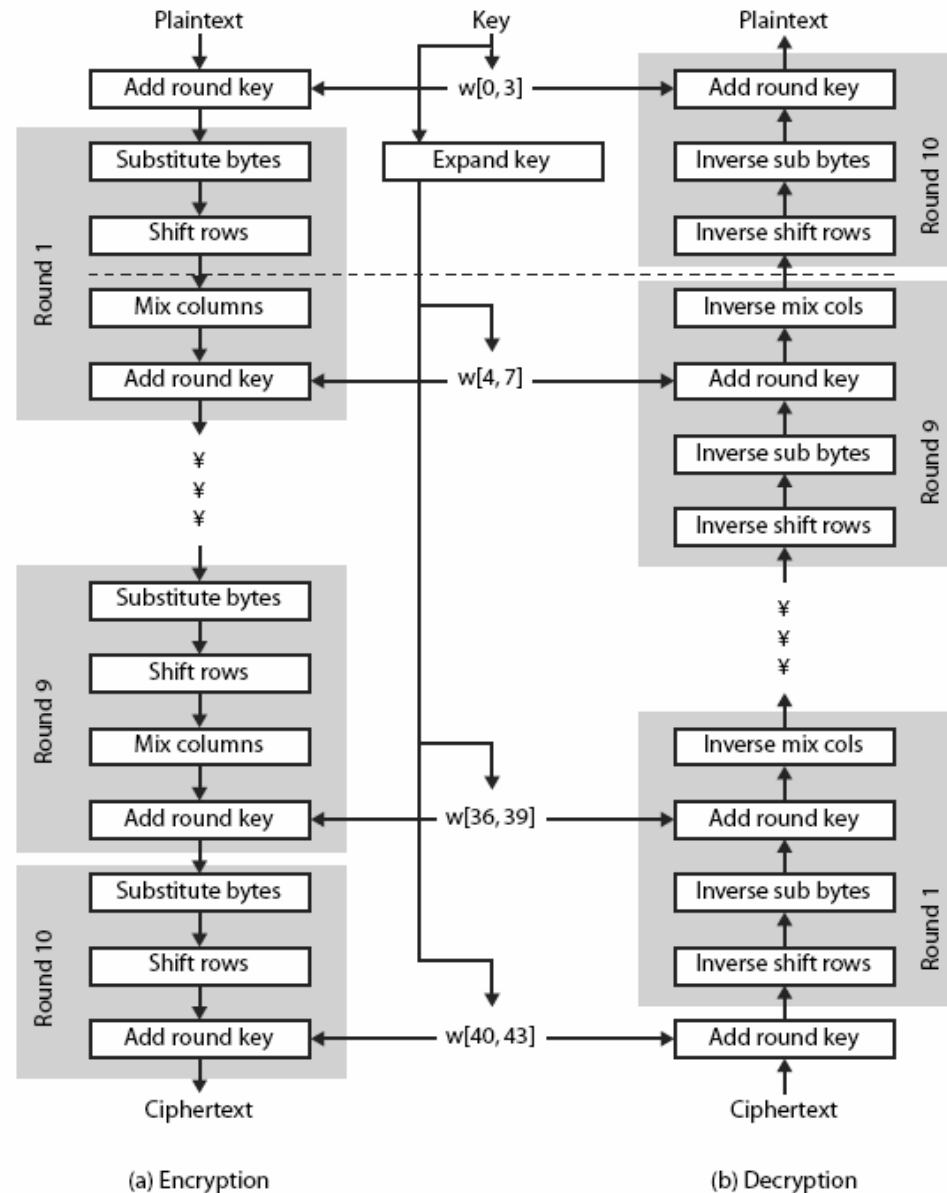
Az AES titkosító - Rijndael

- tervezői: Joan Rijmen és Vincent Daemen (Belgium)
- a Rijndaelben mind az adatblokk mind a kulcshossz 128-256 bit között 32 bitenként változtatható
- AES szabványos kulcsok: 128/192/256 bit,
- AES szabványos adatblokk 128 bit
- **iteratív** S-P hálózat, de nem **Feistel** struktúráss
 - az adatokat 4 sor x 4 oszlopos adatblokkokban tárolja, melyekben egy-egy bájtot tárol
 - minden körben az egész adatblokkon dolgozik
- úgy tervezték, hogy:
 - minden ismert támadásnak ellenálljon
 - gyors és tömör kód írható rá sok processzor típuson
 - átlátható terve legyen

Az AES

- az 4 x 4 bájtós adatblokk egy tartalmát **állapotnak** (state) hívjuk = 4 db 32 bites **szó** = 128 **bit**
- a kulcsot is 32 bites szavak oszlopaira bontjuk ki melyekből körönként szintén 4 db-ot használunk fel
- (kulcsméret függvényében) 10/12/14 körben:
 - byte substitution: (1 közös S-dobozos a bájlthelyettesítés)
 - shift rows: (a sorok elforgatása különböző mértékben)
 - mix columns: (az oszlopok átalakítása mátrixszorzással)
 - add round key: (XOR művelet bájtonként a körkulccsal)
- az első kör előtt még egy **add round key** van
- az utolsó körben a **mix columns** kimarad
- könnyen invertálható és implementálható XOR-ok és műveleti táblázatok segítségével

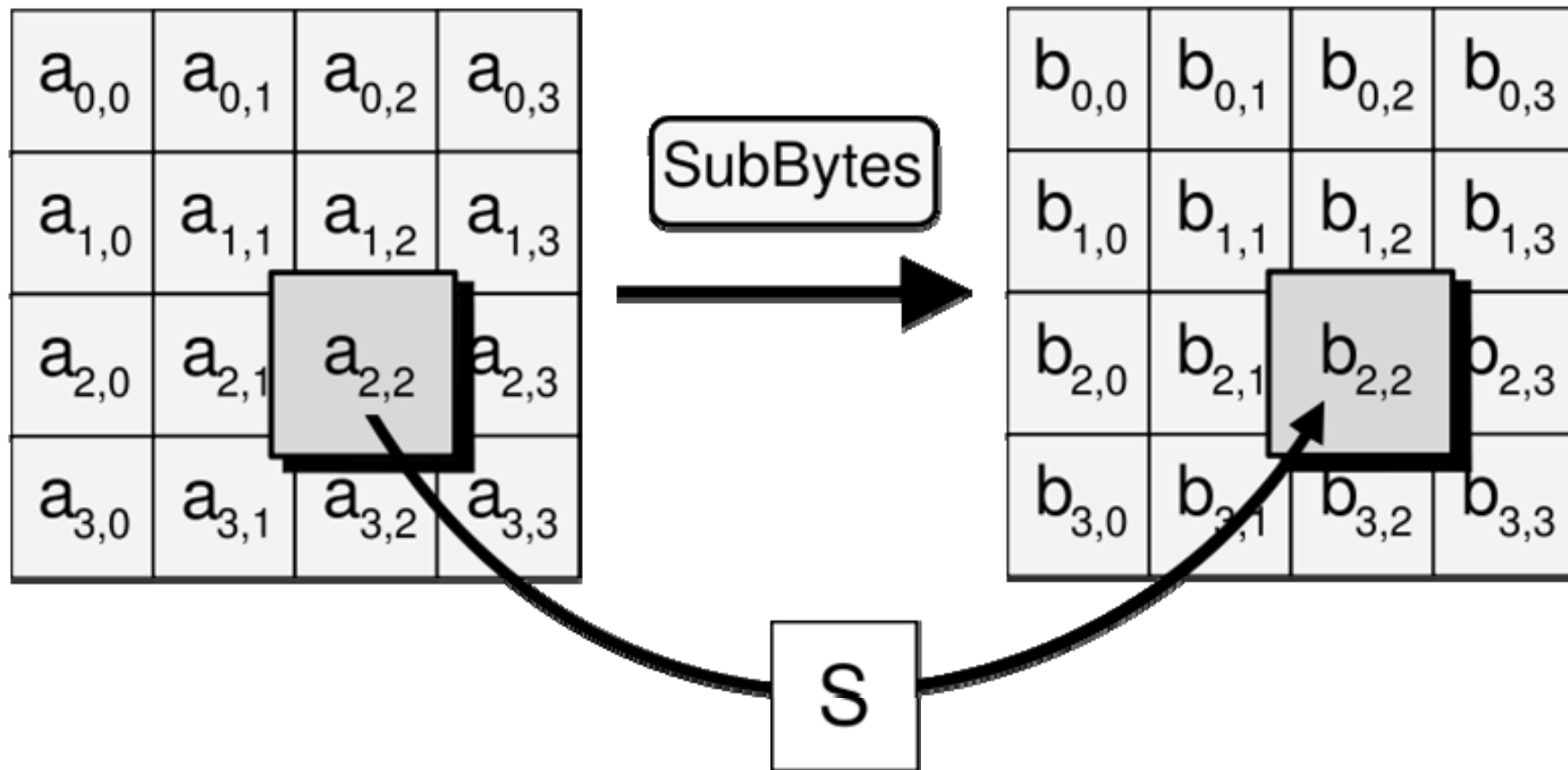
AES titkosítás/megfejtés



A titkosítás algoritmus

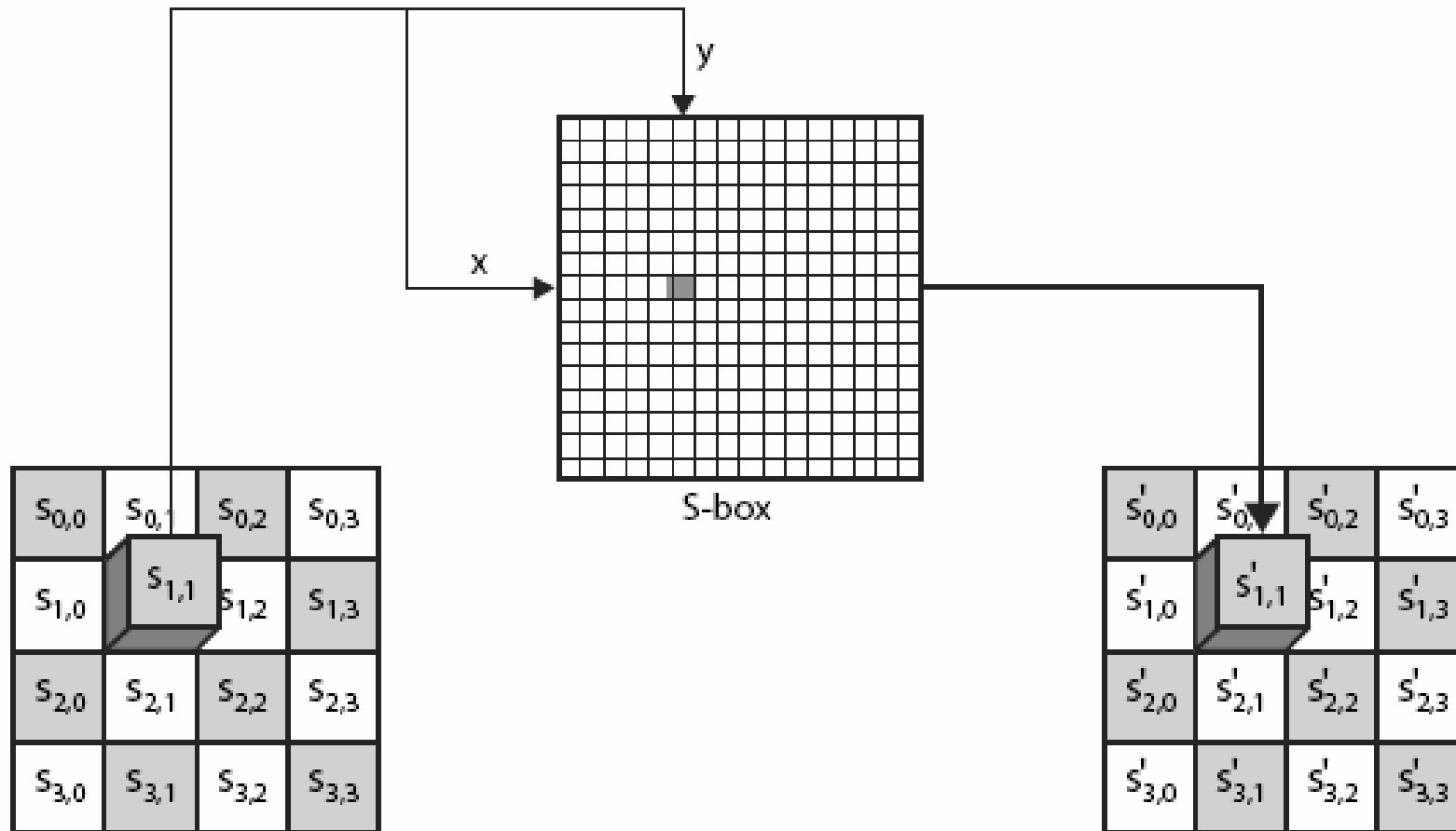
```
AESCipher(byte in[16], byte out[16], word w[44])  
begin  
    byte state[4,4]  
    state = in  
    AddRoundKey(state, w[0, 3])  
    for round = 1 step 1 to 10  
        SubBytes(state)  
        ShiftRows(state)  
        MixColumns(state)  
        AddRoundKey(state, w[round*4,(round+1)*4-1])  
    end for  
    SubBytes(state)  
    ShiftRows(state)  
    AddRoundKey(state, w[40, 43])  
    out = state  
end
```

1. SubBytes() (bájt helyettesítés)



Minden bájt helyettesítése egy közös (8 bit $\rightarrow$ 8 bites)
S-doboz segítségével.

1. SubBytes() megadása táblázattal



Az AES S-doboza táblázatban

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Figure 7. S-box: substitution values for the byte xy (in hexadecimal format).

1. SubBytes()

- egyszerű helyettesítés az aktuális állapot minden bájtjára
- egy 16x16-os bájtokat tartalmazó 256 elemű táblázattal (minden 8-bites érték pontosan egyszer fordul elő)
- a helyettesítendő bájt első 4 bitje a sorindexet a második 4 bitje az oszlopindexet adja
- pl. {95}-öt (hexadecimálisan írva) a 9-es sor 5-ös oszlopában levő {2A} bájjal helyettesítjük
- gondosan úgy tervezték, hogy az ismert támadástípusoknak ellenálljon, pl. „nagyon” nem lineáris
- az S-doboz egy a 256 elemű test feletti invertálható függvénnnyel adható meg
- ez **multiplikatív inverz számítás** a testben, majd **egy affin transzformáció** végrehajtása
- a 256 elemű test elemei polinomok segítségével írhatók le, pl:
$$\text{GF}(2^8) = \mathbf{Z}_2[x] / (x^8+x^4+x^3+x+1)$$

Számolás GF(2⁸)-ban

➤ az együtthatókra $1+1 = 1 \oplus 1 = 0$

➤ és $(x^8+x^4+x^3+x+1)$ -ra nézve redukálni kell. Pl:

$$2d = 00101101 = x^5 + x^3 + x^2 + 1$$

$$a3 = 10100011 = x^7 + x^5 + x + 1$$

$$2d + a3 = x^7 + 2x^5 + x^3 + x^2 + x + 2 = x^7 + x^3 + x^2 + x = 10001110 = 8e$$

$$\begin{aligned} 2d \bullet a3 &= (x^{12} + x^{10} + x^9 + x^7) + (x^{10} + x^8 + x^7 + x^5) \\ &\quad + (x^6 + x^4 + x^3 + x) + (x^5 + x^3 + x^2 + 1) \\ &= x^{12} + x^9 + x^8 + x^6 + x^4 + x^2 + x + 1 = \end{aligned}$$

$$\begin{aligned} &= (x^4 + x) \cdot (x^8 + x^4 + x^3 + x + 1) + x^7 + x^6 + x^4 + 1 = \\ &= x^7 + x^6 + x^4 + 1. \end{aligned}$$

$$\text{Így } 2d \bullet a3 = 11010001 = d1$$

➤ Ez a struktúra test, mert bebizonyítható, hogy minden nem 0 polinomnak van inverze.

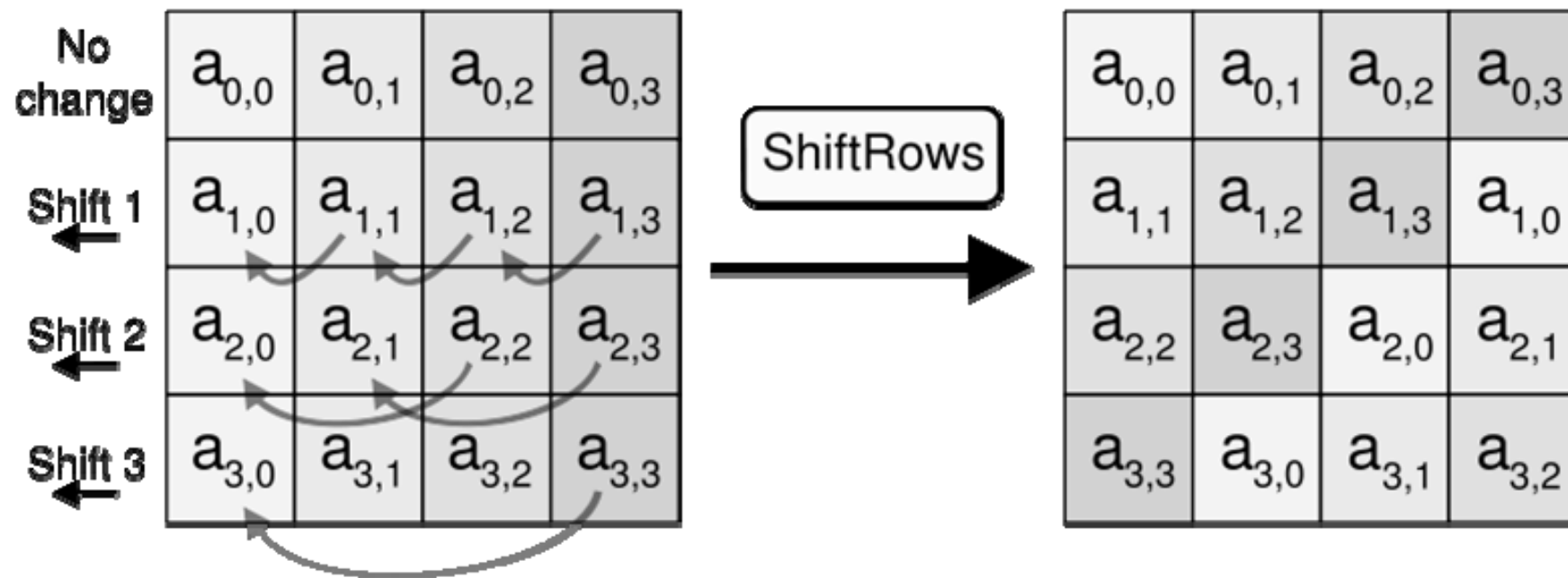
Az S-doboz megkonstruálása

- bemenet: $(a_7, a_6, \dots, a_0)$ 8 bit,
- inverze $G(256)$ -ban legyen: $(x_7, x_6, \dots, x_0)$
- majd egy affin transzformáció:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

- Másképpen bitenként kiszámolva:
- $b_i = x_i \oplus x_{i+4 \bmod 8} \oplus x_{i+5 \bmod 8} \oplus x_{i+6 \bmod 8} \oplus x_{i+7 \bmod 8} \oplus c_i$
ahol a b_i kimenet i -dik bitje, és $c = 01100011$.
- kimenet: $(b_7, b_6, \dots, b_0)$ 8 bit.

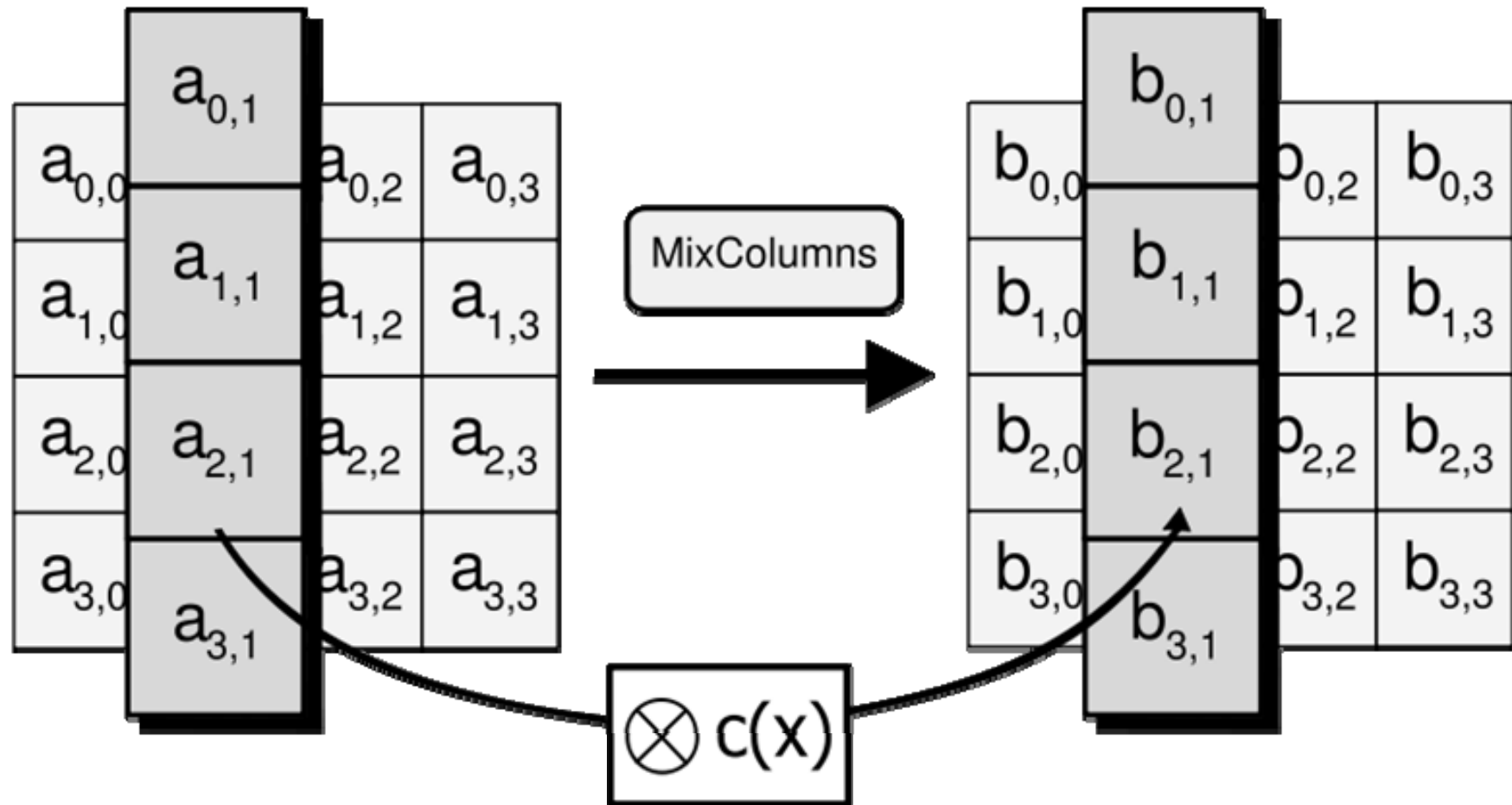
2. Shift Rows()



2. ShiftRows() (sorok elforgatása)

- permutációs lépés (keverés)
- a bájtok körkörös forgatása balra
 - az 1. sor változatlan marad
 - a 2. sor 1 bájttal fordul körbe balra
 - a 3. sor 2 bájttal fordul körbe balra
 - a 4. sor 3 bájttal fordul körbe balra
- a megfejtéskor egyszerűen jobbra forgatunk
- mivel az állapotokban a bájtokat oszloponként tekinthetjük, ez a lépés az oszlopok adatait keveri össze
- minden oszlop kap bájtot minden oszlopból

3. MixColumns()

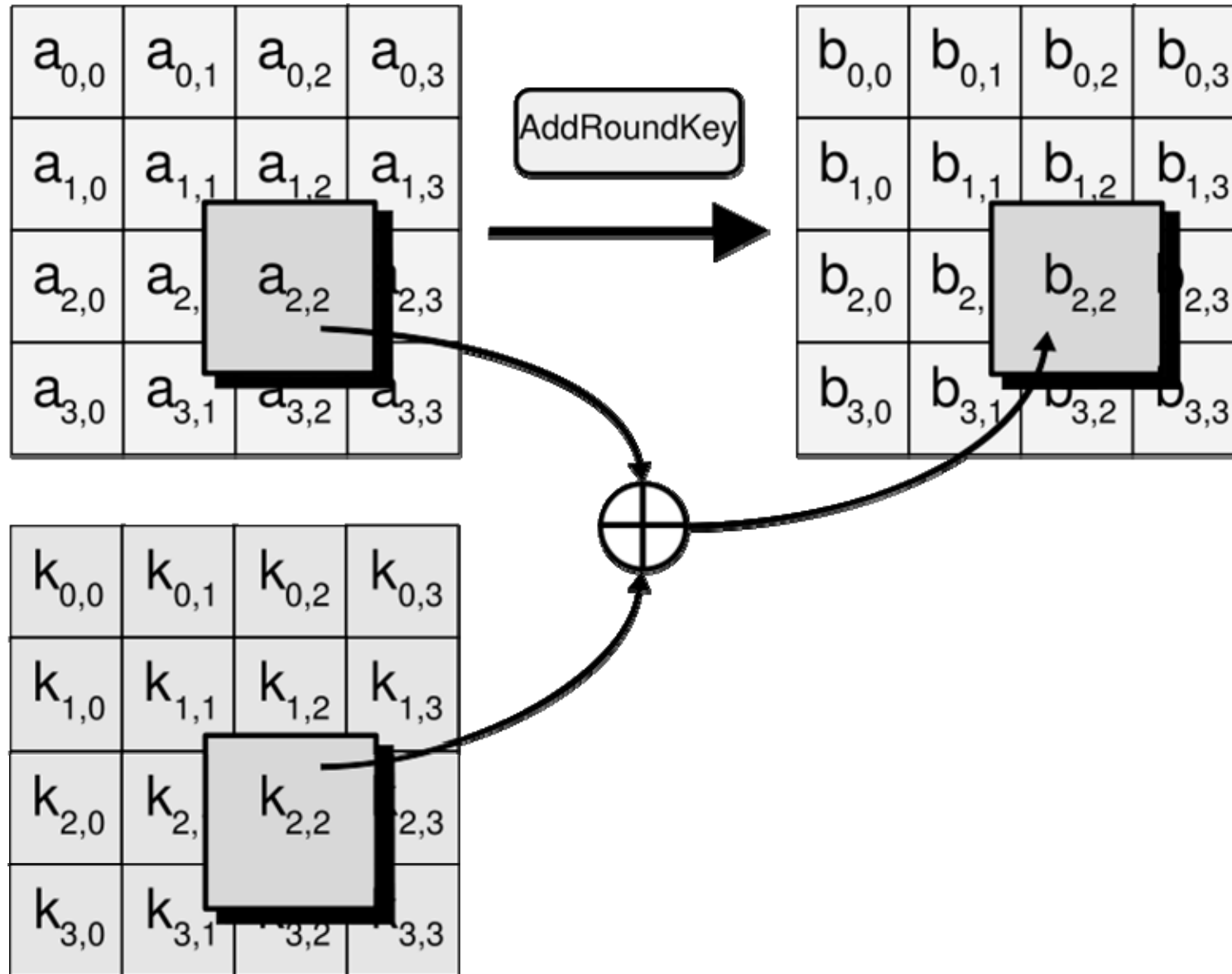


3. MixColumns() (oszlopok összekeverése)

- az oszlopokat külön-külön helyettesítjük
- minden helyettesített bájt a keverendő oszlop mind a 4 bájtjának függvénye
- effektíven leírható egy mátrix szorzással a 256 elemű $GF(2^8)$ test felett, ahol a bájtok szorzása polinomok segítségével történő bonyolult, de könnyen implementálható művelet:

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

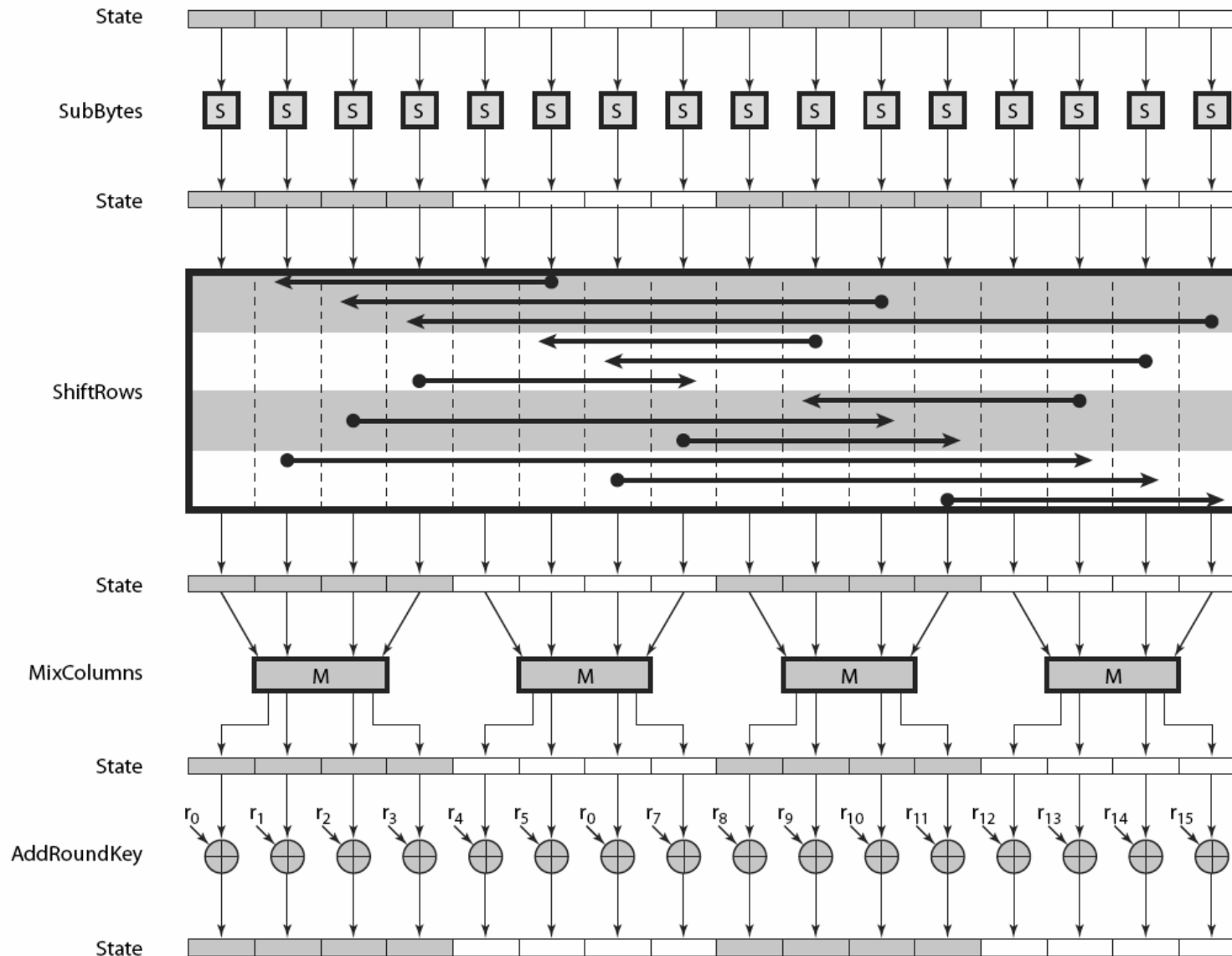
4. AddRoundKey()



4. AddRoundKey() (körkulcs hozzáadása)

- annyira egyszerű, amennyire csak lehet
- XOR az állapot és a 128 bites körkulcs között
- szintén oszloponként haladva (persze bájt szinten párhuzamosítható)
- inverze a megfejtéskor ugyanez
- mivel a XOR önmaga inverze, csak a kulcsok sorrendjét kell megfordítani

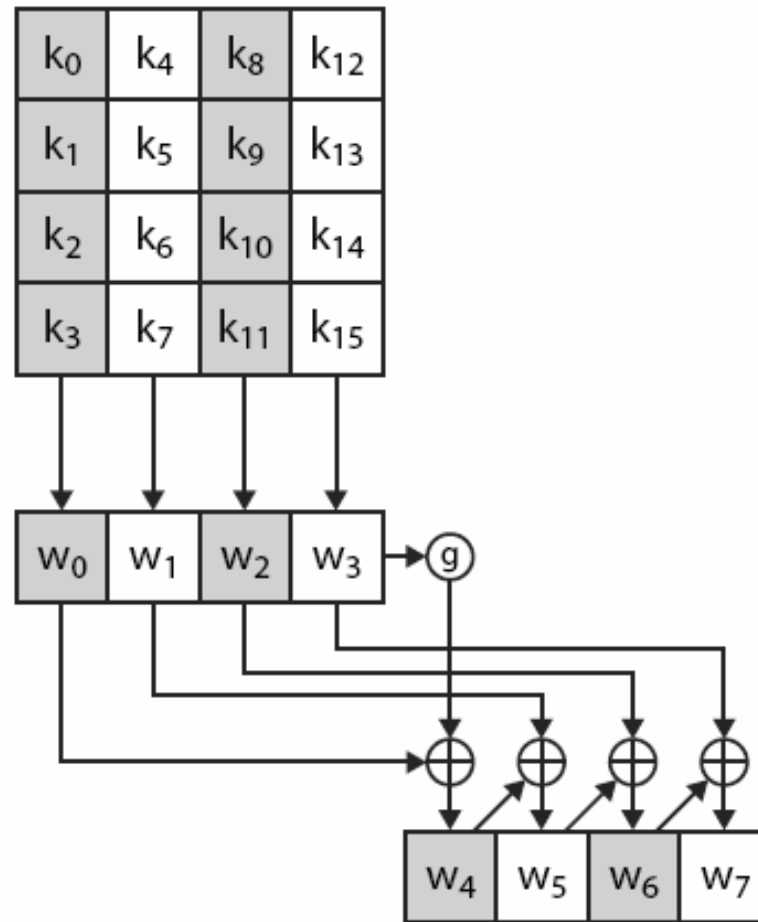
Egy AES kör az állapotok „oszloponkénti kiterítésével”



AES kulcs kiterjesztés (Key Expansion)

- minden körhöz 4 (32-bites) szóból álló körkulcs kell
- a körök számától függően egy összesen 44/52/60 szóból álló kulcstömbre van szükség
- a kulcstömb első 4 eleme maga a titkosítás kulcsa (128 bit)
 W_0, W_1, W_2, W_3
- a kulcstömb további szavait ezek kiterjesztésével kapjuk
- a következő szót mindig **az előzőből** és a 4-gyel ezelőttiből képezve
 - 4ből 3 esetben egyszerűen XOR-olva a kettőt
 - minden 4. esetben viszont (ha az új w_i indexe 4-gyel osztható) az előző szót előbb
 1. elforgatjuk
 2. az S-doboz alapján helyettesítjük
 3. egy kör konstanssal XOR-oljukmielőtt a 4-el ezelőttivel XOR-olnánk

AES kulcs kiterjesztés



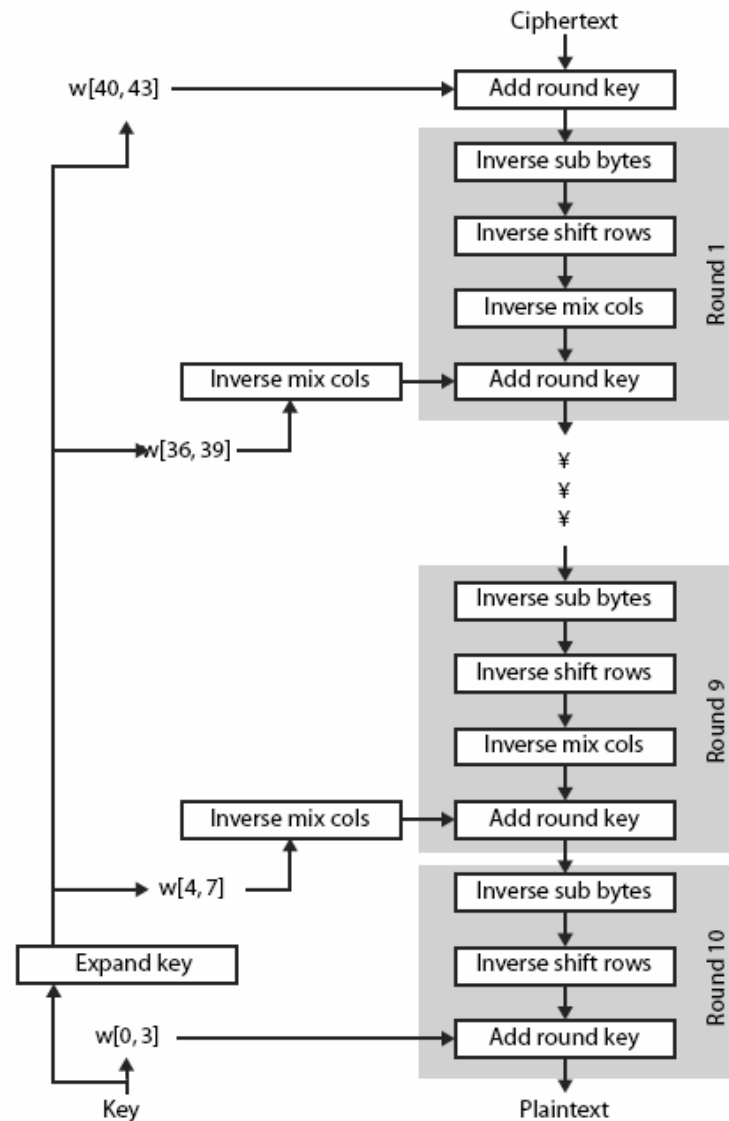
A kulcs kiterjesztés magyarázata

- hogy az ismert támadásoknak ellenálljon
- a következő tervezési kritériumok vették:
 - a kulcs egy részének ismeret ne segítse a többi bit meghatározását
 - a transzformáció invertálható legyen
 - gyorsan működjön különböző a processzorokon
 - a körkonstansok alkalmazása megtörje a szimmetriát
 - biztosítsa a kulcs biteinek diffúzióját a körkulcsok biteibe
 - küszöbölje ki a linearitást, így gátolva az analízist
 - egyszerűen leírható/megvalósítható legyen

Az AES megfejtés

- mind a 4 lépésnek van inverze: pl. InvSubBytes()
ezek megegyeznek az eredetivel csak más konstansokkal/táblázatokkal dolgoznak
- az AES megfejtés nem egyezik meg a titkosítással mivel a **lépések inverzét fordított sorrendben kell alkalmazni**
- de szerencsére lehetőség van a titkosítással megegyező struktúrájú ekvivalens megfejtő algoritmus kidolgozására
 - természetesen fordított kulcssorrenddel
 - a lépések inverzeihez tartozó más táblázatokkal
- erre azért van lehetőség, mert
 - InvSubBytes és InvShiftRows sorrenje felcserélhető
 - InvMixColumns és AddRoundKey is felcserélhető, ha a körkulcson előbb InvMixColumns-t elvégezzük, hiszen InvMixColumns lineáris transzformáció.

AES ekvivalens megfejtés



Implementációs szempontok I

- effektíven implementálható 8-bites CPU-ra:
 - SubBytes leírható egy 256 elemű bájtokat tartalmazó táblázattal
 - ShiftRows egyszerűen bájtok cseréje
 - AddRoundKey bájtonkénti XOR művelet
 - MixColumns ugyan mátrixszorzás $GF(2^8)$ felett, de egyszerűsíthető egy másik 256 elemű, bájt értéket tartalmazó tömbben való keresések és bájtonkénti XOR műveletek használatára
 - csak {02}-vel kell tudni szorozni, mert $\{03\} = \{02\} \oplus \{01\}$

Implementációs szempontok II

- effektíven implementálható 32-bites CPU-ra is:
 - a kör 4 függvényét összevonhatjuk 32-bites oszlopok közvetlen kiszámítására táblázatokkal
 - ehhez előre kiszámíthatunk 4 darab 256 bemenetű 32-bites szó értékeket tartalmazó táblázatot
 - egy eredmény oszlop kiszámítható 4 darab kereséssel a táblázatokban + 4 szavankénti XOR-ral
 - a táblázatok mérete összesen $4 \times 256 \times 4 = 4096$ bájt
 - ha nincs 4KB hely megoldható 1 táblázattal és ciklikus forgatásokkal is
- a tervezők szerint ez az effektív megvalósíthatóság kulcsfontosságú tényező volt abban, hogy a Rijndaelt választották AES titkosítónak

Az AES biztonsága

- a módszer feltörésének minősül minden olyan módszer, ami pl. a teljes kipróbálás átlagos 2^{127} titkosítási műveleténél kevesebbrel fejt meg a 128 bites kulcsot
- mondjuk egy 2^{120} műveletigényű, 2^{100} választott nyíltszöveget igénylő módszer is, ami a gyakorlatban biztos, hogy kivitelezhetetlen
- jelenleg (2008-ban) **nem ismert AES törés** csupán az AES kevesebb körrel rendelkező változataira van ilyen:
 - ismert nyílt szövegű támadás
 - 7 körös AES-128-ra
 - 8 körös AES-192-re és AES-256-ra
 - hasonló kulcsos támadás (related key attack)
 - 9 körös AES-256-ra

Az XSL támadás (?)

- 2002-ben ugyan napvilágot látott egy spekulatív „XSL attack” nevezetű támadás
- de még nyitott kérdéses, hogy valóban alkalmazható-e elméletileg is az AES-re
- jelenleg gyakorlati kivitelezése nagyon valószínűtlen

Kerülő utas támadások (**side channel attacks**)

- ezek nem az algoritmust, hanem annak implementációját támadják
- nevezetesen **cache timing attacks** (gyorsítótár időmérési támadások)
- de kivitelezésükhöz a titkosító rendszer nagyon pontos megfigyelésére van szükség,
- pl. a titkosító szerveren futó időmérő programra, vagy 200 millió választott nyílt szövegre
- komoly gyakorlati fenyegetést nem jelentenek
- sajnos nagyon nehéz olyan implementációt írni, hogy a táblázatokban való keresés gyors, de az indextől teljesen független legyen

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#> (3.4.-3.5. fejezet)
- William Stallings: Cryptography and Network Security, 4th Edition, Prentice Hall, 2006. (Chapter 5)
- Lawrie Brown előadás fóliái (Chapter 5)
- [http://en.wikipedia.org/wiki/Advanced Encryption Standard](http://en.wikipedia.org/wiki/Advanced_Encryption_Standard)
- AES animáció: http://www.cs.bc.edu/~straubin/cs381-05/blockciphers/rijndael_ingles2004.swf