Department of Electronics

Cryptography

Fall 2019

Hasan Mahmood

hasan@qau.edu.pk

Week 6 (09 & 10 October 2019)

Modes of operation

- DES is a block cipher, 64-bits blocks, longer or shorter messages
- Character by character transmission (messages shorter than 64-bits)
- Many modes of operation, allowing users to choose appropriate modes to meet the requirements of their applications

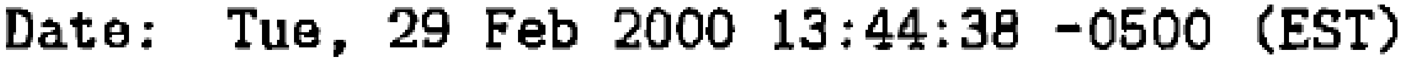
1. Electronics codebook (ECB)
2. Cipher block chaining (CBC)
3. Cipher feedback (CFB)
4. Output feedback (OFB)
5. Counter (CTR)

Electronics Codebook (ECB)

- Break plaintext into appropriate sized blocks, and process separately
- Encryption function E_k is used
- This is know as the electronics codebook (ECB) mode of operation
- Plaintext: $P=[P_1, P_2, P_3, …., P_L]$
- Cyphertext : $C=[C_1, C_2,…., C_L]$
- Where $C_j=E_K(P_j)$ *is the encryption of* P_j *using key* K
- Apparent weakness when plaintext is long

ECB weakness

- Eve has been observing communication between Alice and Bob for long enough period of time
- If Eve has managed to acquire some plaintext pieces corresponding to the ciphertext pieces (that was observed)
- Eve can start to build up a codebook with which Eve can decipher future communication between Alice an Bob
- Eve never needs to calculate the Key
- Codebook is used to decipher the communication
- Real problem if the fragments are repeated in the plaintext
- Email header example, it repeats on specific dates
- False ciphertext message corrupt the original message



```
Date:  Tue, 29 Feb 2000 13:44:38 -0500 (EST)
```

Cipher Block Chaining (CBC)

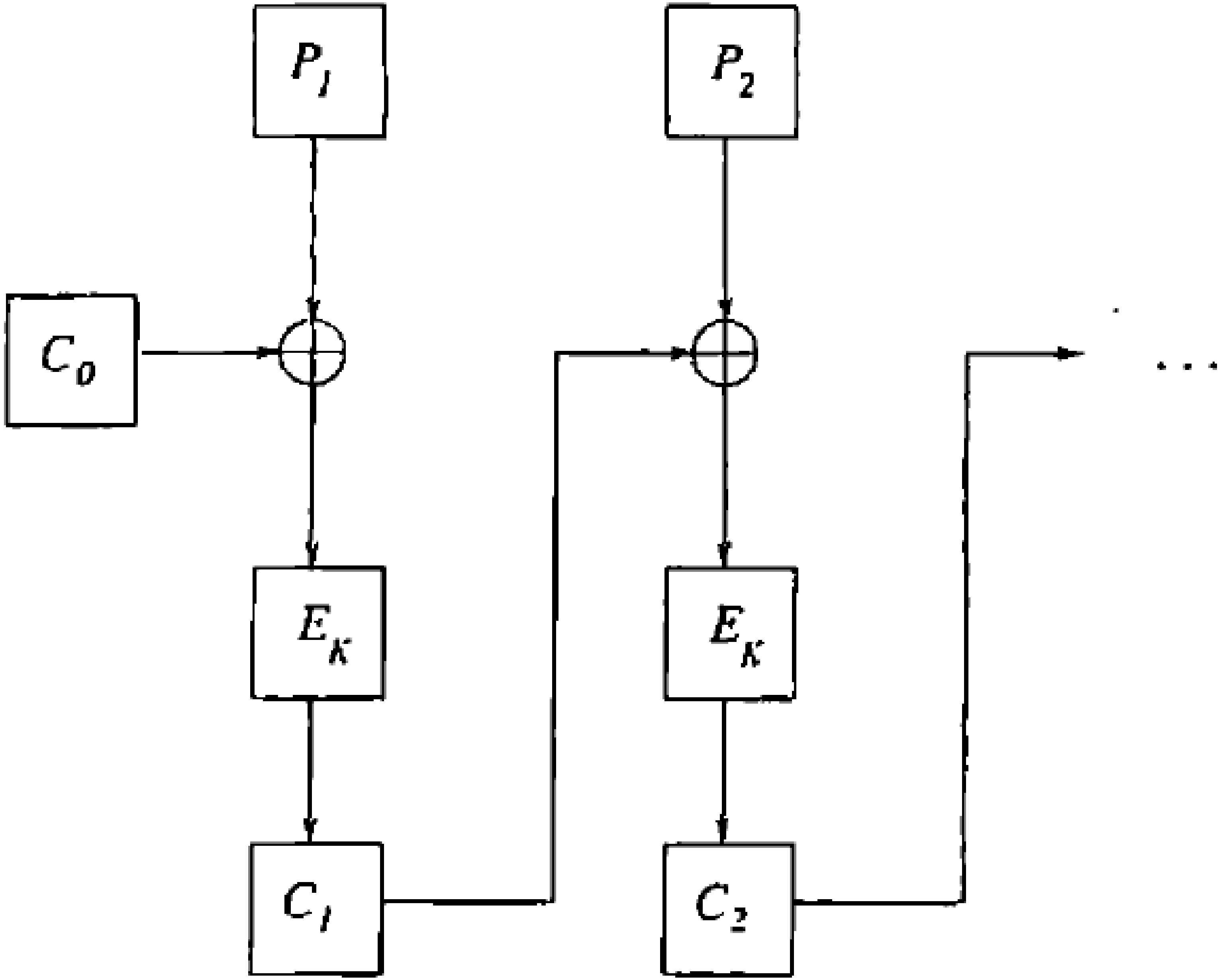
- Reduce problem in ECB mode is to use chaining
- Chaining is a feedback mechanism where the encryption of block depends on the encryption of the previous blocks
- In general, encryption proceeds as

$$C_j = E_K(P_j \oplus C_{j-1}),$$

- With decryption as
-

$$P_j = D_K(C_j) \oplus C_{j-1}$$

CBC



Cipher Feedback (CFB)

- CBC and ECB work when complete block of 64-bit of plaintext is available
- Based on Linear Feedback Shift Register (LFSR)
- Cipher feedback mode is a stream mode of operation that produces pseudorandom bits using the block cipher E_K
- In general, it operates in a *k*-bit mode, where each application produces *k* random bits XORing with the plaintext (8-bit version)
- Useful for interactive computer applications
- Plaintext is broken into 8-bit pieces $P=[P_1, P_2, \ldots]$

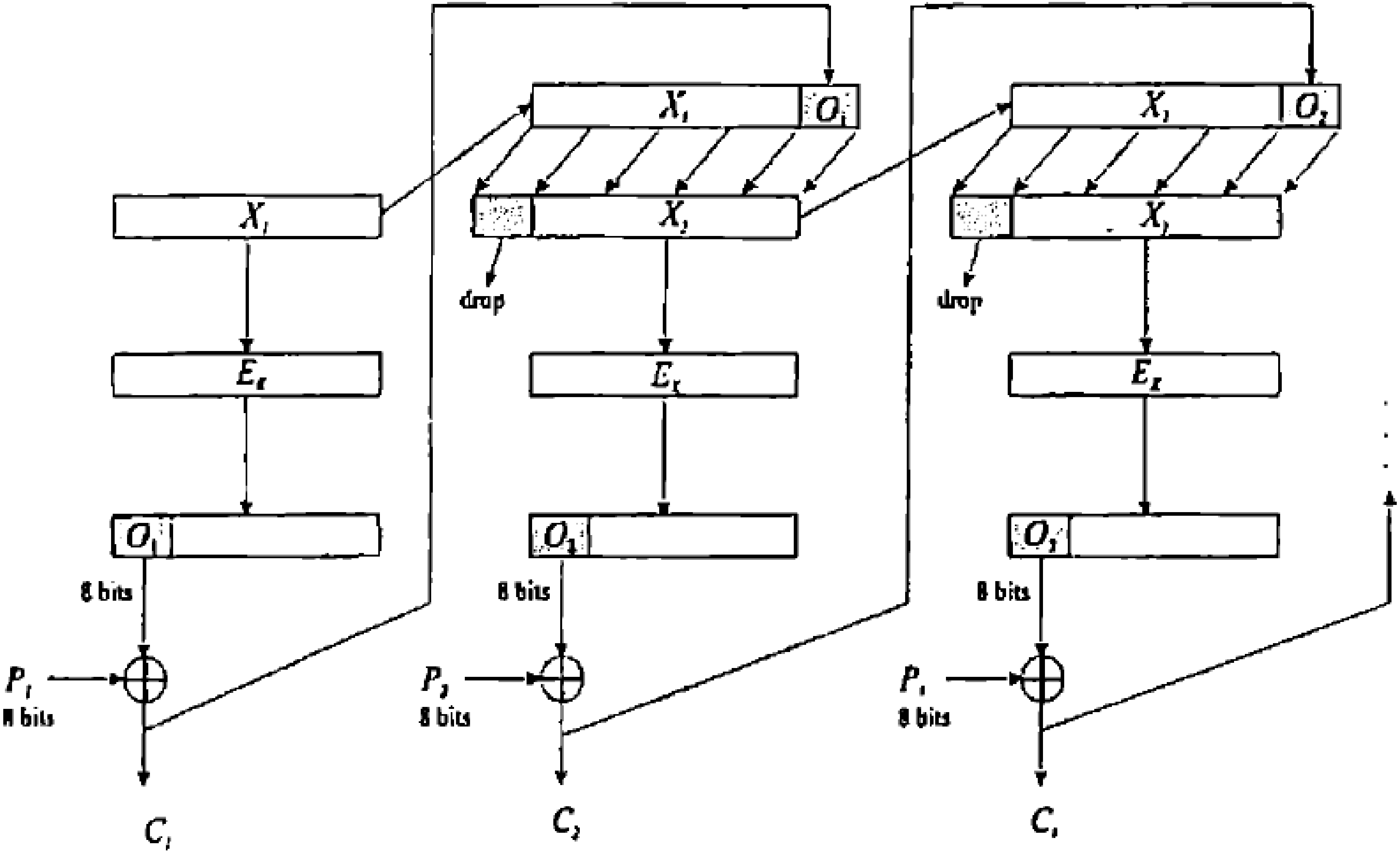
Cipher Feedback (CFB) Encryption

- An initial 64-bit X_1 is chosen, then for j=1,2,3,…. the following is performed:

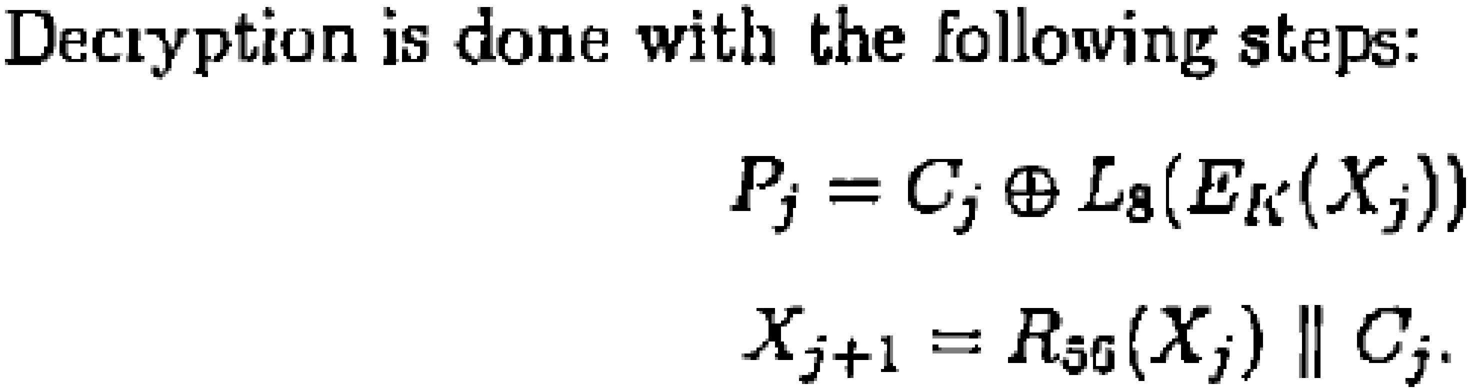
$$\begin{aligned} O_j &= L_8(E_K(X_j)) \\ C_j &= P_j \oplus O_j \\ X_{j+1} &= R_{56}(X_j) \,\|\, C_j, \end{aligned}$$

- $L_8(X)$ denotes the 8 leftmost bits of X
- $R_{56}(X)$ denotes the rightmost 56 bits of X
- $X||Y$ denotes the string obtained by wiring X followed by Y

CFM


X1
O1
X2
O2
X1
drop
X2
drop
X3
EK
EK
EK
O1
O2
O3
8 bits
8 bits
8 bits
P1
8 bits
P2
8 bits
P3
8 bits
C1
C2
C3

CFB decryption



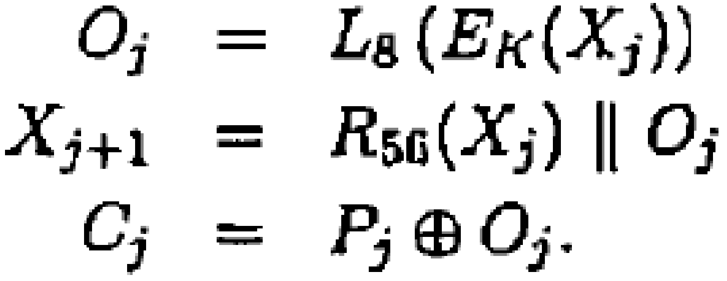
Decryption is done with the following steps:

$$P_j = C_j \oplus L_8(E_K(X_j))$$

$$X_{j+1} = R_{56}(X_j) \parallel C_j.$$

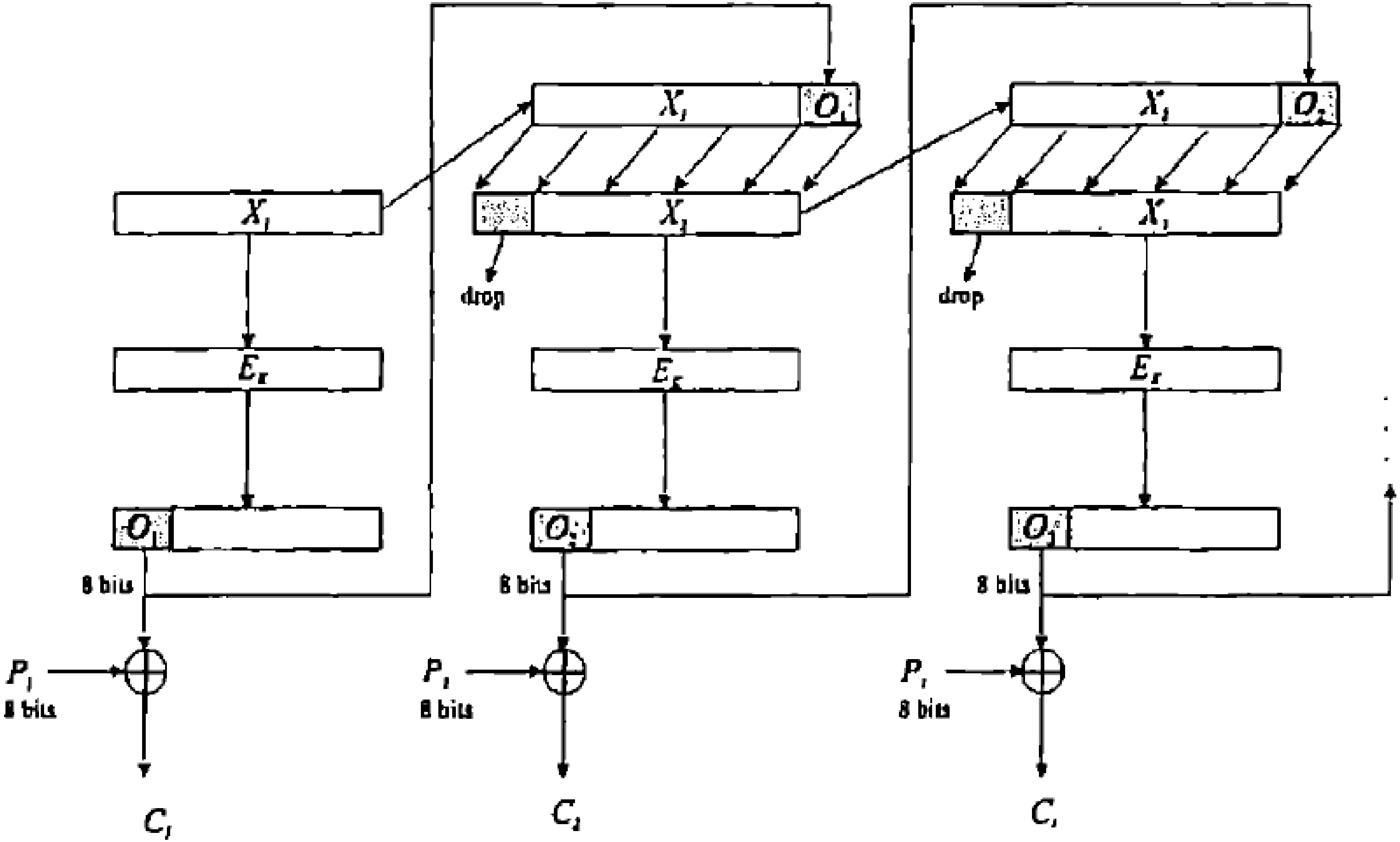
Output Feedback (OFB)

- CBC and CFB modes of operation exhibit a drawback in that errors prorogate for a duration of time corresponding to the block size
- Stream cipher, XORing the message with a pseudo0random bit stream generation by the block cipher
- It avoids error propagation



$$\begin{aligned} O_j &= L_8(E_K(X_j)) \\ X_{j+1} &= R_{56}(X_j) \parallel O_j \\ C_j &= P_j \oplus O_j. \end{aligned}$$

OFB

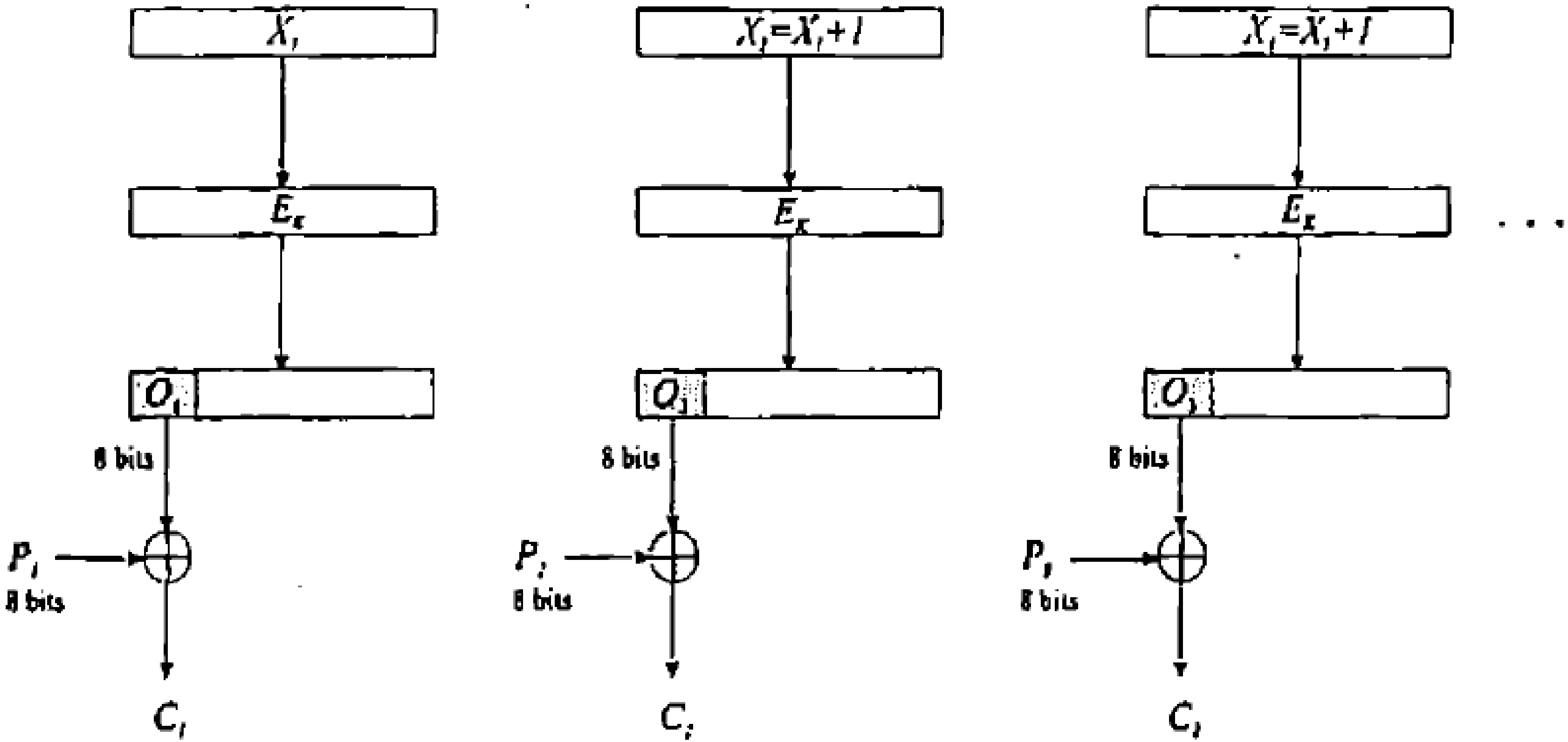

X_1
E_K
O_1
8 bits
P_1
8 bits
C_1
X_1
O_1
X_2
drop
E_K
O_2
8 bits
P_2
8 bits
C_2
X_2
O_2
X_3
drop
E_K
O_3
8 bits
P_3
8 bits
C_3

Counter (CTR)

- Based on the ideas that were used in OFB mode
- Creates output key stream that is XORed with chunks of plaintext to produce ciphertext

$$\begin{aligned} X_j &= X_{j-1} + 1 \\ O_j &= L_8\left(E_K(X_j)\right) \\ C_j &= P_j \oplus O_j \end{aligned}$$

CTR



X_i
X_i=X_i+1
X_i=X_i+1
E_K
E_K
E_K
O_i
O_i
O_i
8 bits
8 bits
8 bits
P_i
8 bits
P_i
8 bits
P_i
8 bits
C_i
C_i
C_i
. . .

The Advanced Encryption Standard: Rijndael

- In 1997, NIST CFP to replace DES (15 proposals submitted, 5 finalist)
- Key sizes 128, 192 and 256 on blocks of 128 bits
- Not computationally complex, 8-bit, 16-bit computers etc.
    1. MARS (from IBM)
    2. RC6 (from RSA laboratories)
    3. Rijndael (from Joan Daemen and Vincent Rijmen)
    4. Serpent (from Ross Anderson, Eli Biham, and Lars Kundsen)
    5. Twofish (from Bruce Schneier, John Kelsey, Doung Whiting, David Wagner, Chris Hall and Niels Ferguson)
- Rijndael was chosen as the Advanced Encryption Standard

AES Basic algorithm, basic steps

- Modes of operation: ECB, CBC, CFB, OFB and CTR
- 10 rounds (when key is 192/256 bits, 12/14 rounds are used)
- Each round has a round key, derived from the original key
- Four basic steps, called the layers, that are used to form the rounds

1. **The ByteSub Transformation (BS):** This non-linear layer is for resistance to differential and linear cryptanalysis attacks.

2. **The ShiftRow Transformation (SR):** This linear mixing step causes diffusion of the bits over multiple rounds.

3. **The MixColumn Transformation (MC):** This layer has a purpose similar to ShiftRow.

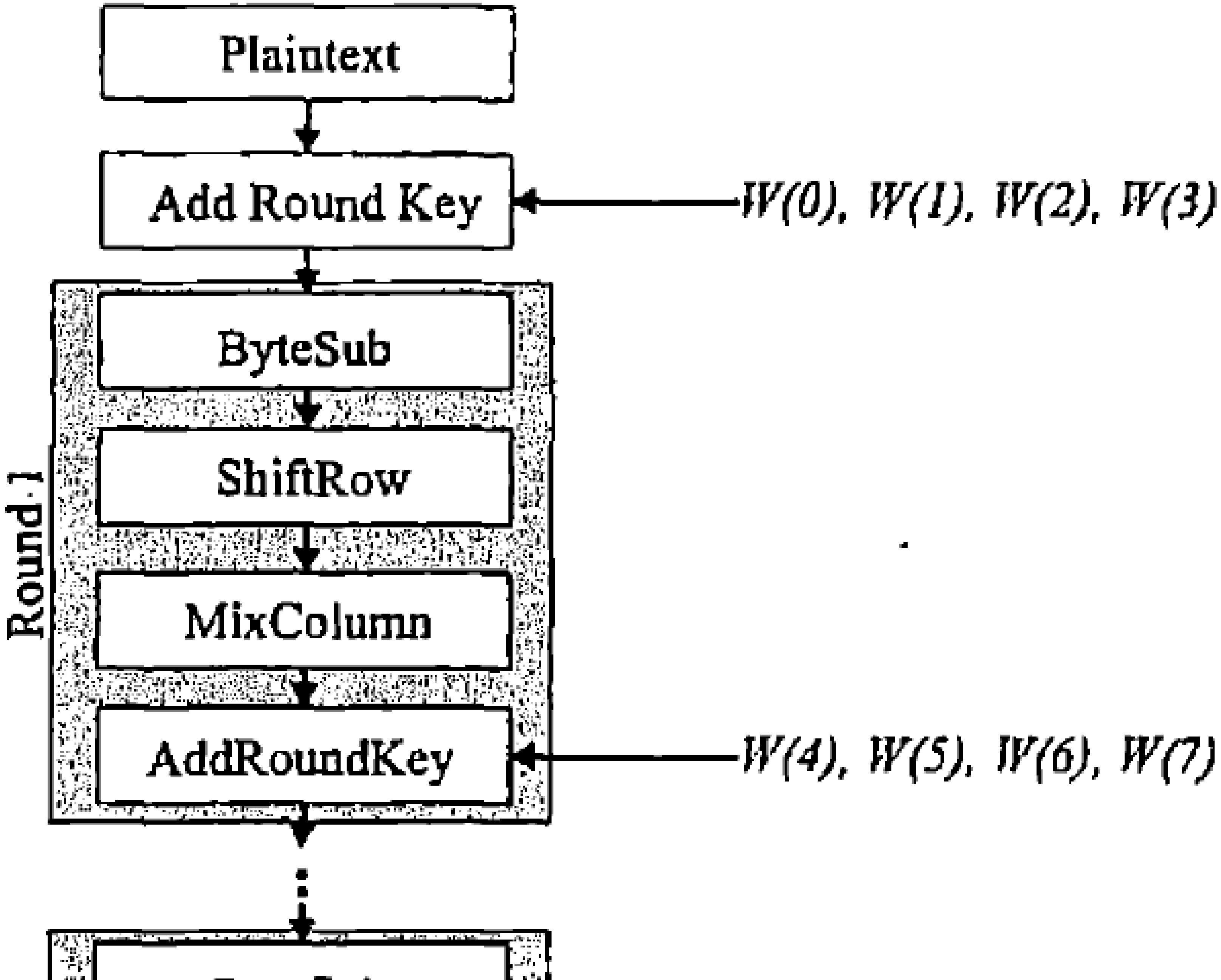
4. **AddRoundKey (ARK):** The round key is *XOR*ed with the result of the above layer.

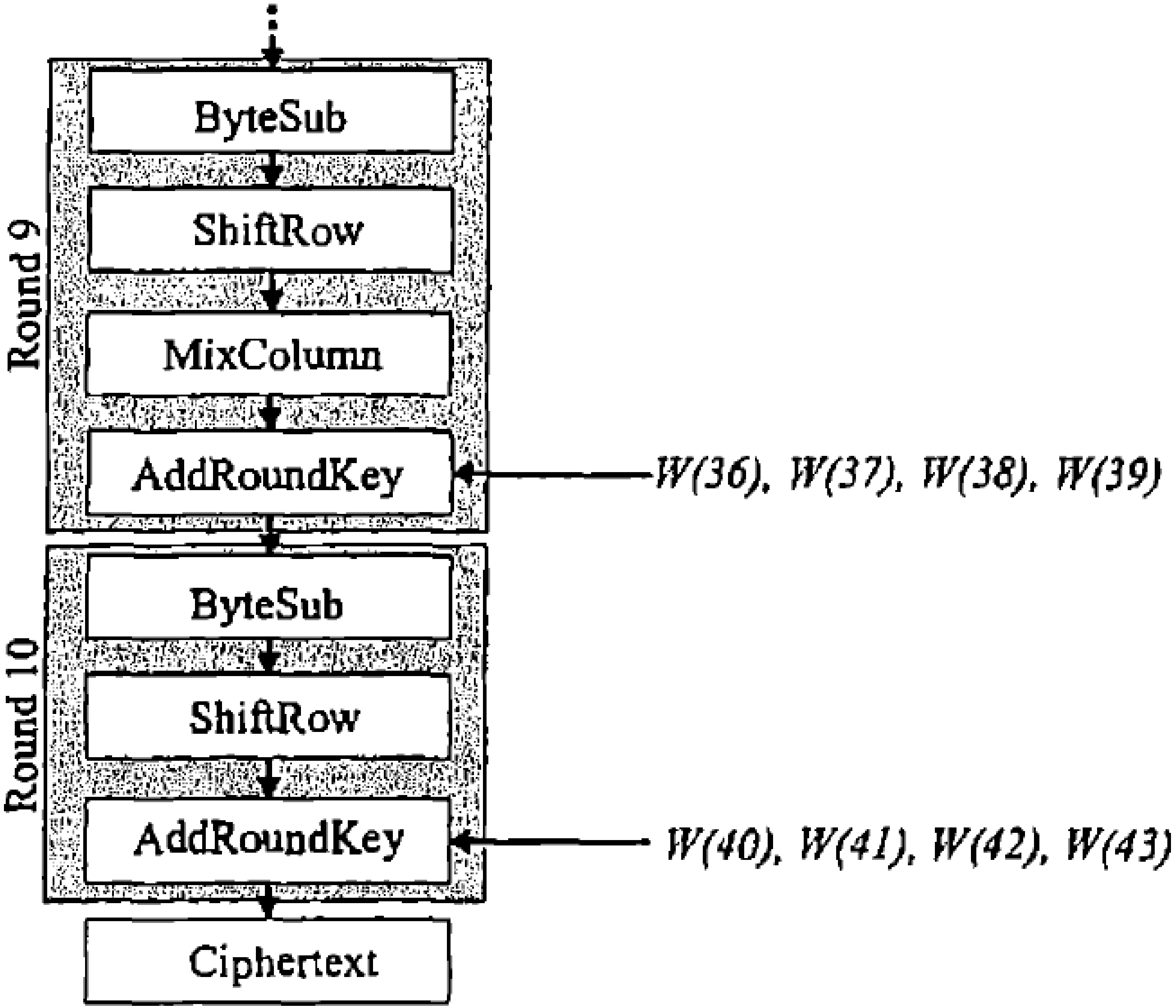
A round is then

$$\rightarrow \boxed{ByteSub} \rightarrow \boxed{ShiftRow} \rightarrow \boxed{MixColumn} \rightarrow \boxed{AddRoundKey} \rightarrow .$$

Rijndael Encryption

1. ARK, using the 0th round key.
2. Nine rounds of BS, SR, MC, ARK, using round keys 1 to 9.
3. A final round: BS, SR, ARK, using the 10th round key.


Plaintext
Add Round Key
W(0), W(1), W(2), W(3)
Round 1
ByteSub
ShiftRow
MixColumn
AddRoundKey
W(4), W(5), W(6), W(7)


Round 9
ByteSub
ShiftRow
MixColumn
AddRoundKey
W(36), W(37), W(38), W(39)
Round 10
ByteSub
ShiftRow
AddRoundKey
W(40), W(41), W(42), W(43)
Ciphertext

AES Layers

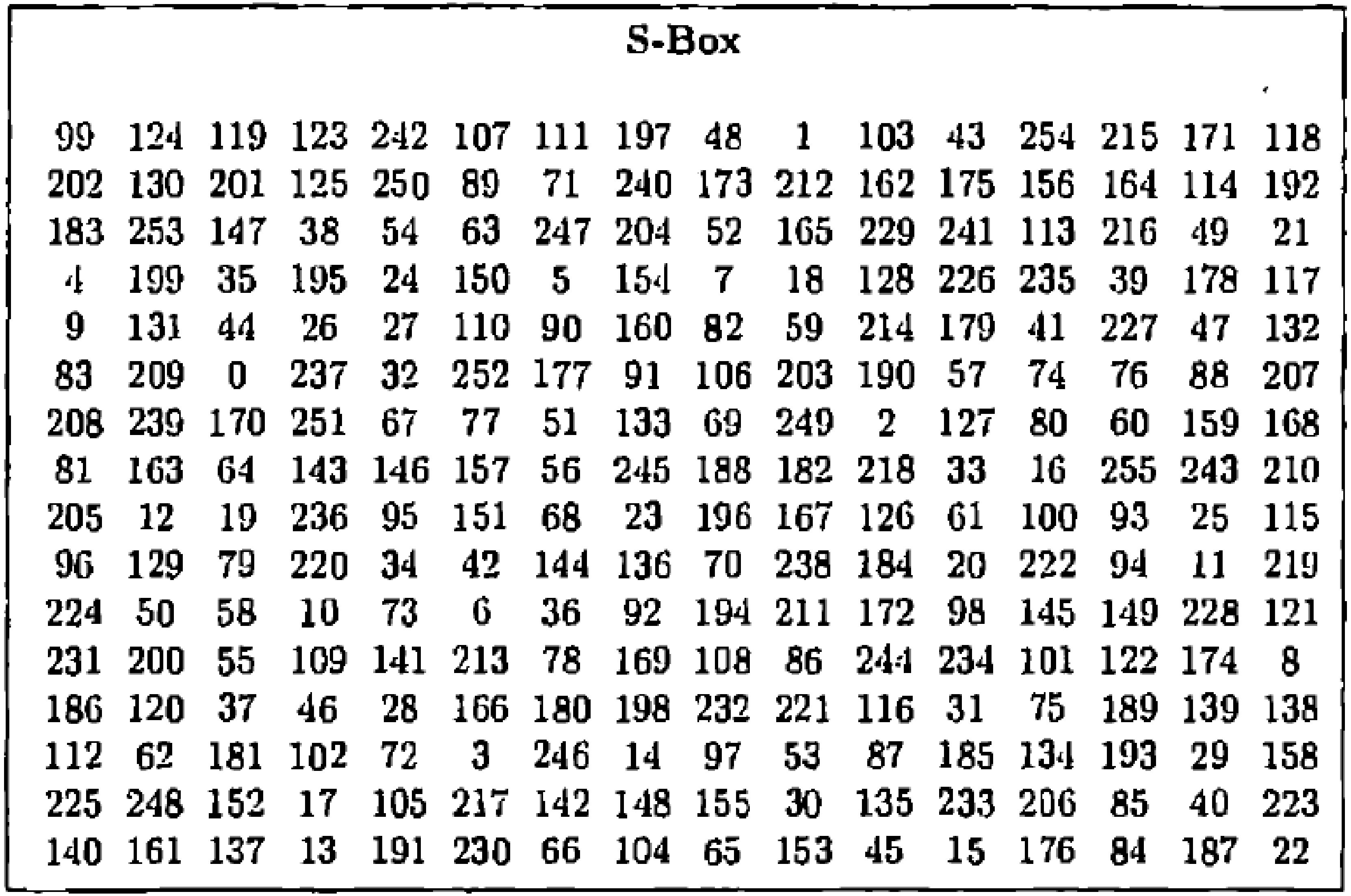
- 128-bits are grouped into 16 bytes of 8-bits each arranged into a matrix:

$$\begin{pmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \end{pmatrix} .$$

- Each byte has a multiplicative inverse, that is, there is a byte *b’* such that *b.b’*=00000001, $GF(2^8)$
- Irreducable polynomial of degree 8, $X^8+X^4+X^3+X+1$

The ByteSub Transformation

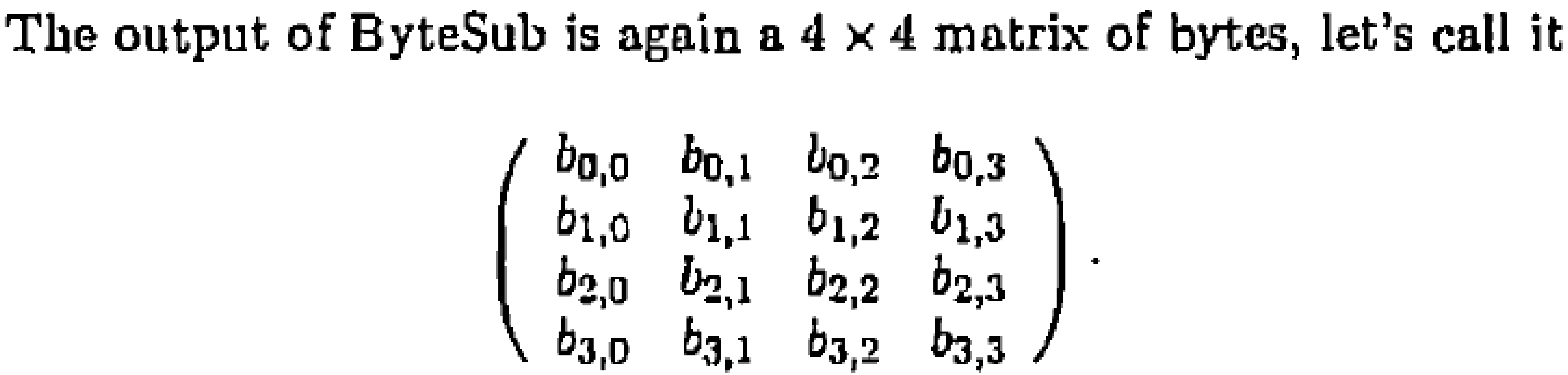
- Each byte in a matrix is changed to another byte by S-box.
- Write a byte as 8-bits: *abcdefgh,* look for the entry in the *abcd* row and *efgh* column (numbered from 0 to 15)



S-Box

99	124	119	123	242	107	111	197	48	1	103	43	254	215	171	118
202	130	201	125	250	89	71	240	173	212	162	175	156	164	114	192
183	253	147	38	54	63	247	204	52	165	229	241	113	216	49	21
4	199	35	195	24	150	5	154	7	18	128	226	235	39	178	117
9	131	44	26	27	110	90	160	82	59	214	179	41	227	47	132
83	209	0	237	32	252	177	91	106	203	190	57	74	76	88	207
208	239	170	251	67	77	51	133	69	249	2	127	80	60	159	168
81	163	64	143	146	157	56	245	188	182	218	33	16	255	243	210
205	12	19	236	95	151	68	23	196	167	126	61	100	93	25	115
96	129	79	220	34	42	144	136	70	238	184	20	222	94	11	219
224	50	58	10	73	6	36	92	194	211	172	98	145	149	228	121
231	200	55	109	141	213	78	169	108	86	244	234	101	122	174	8
186	120	37	46	28	166	180	198	232	221	116	31	75	189	139	138
112	62	181	102	72	3	246	14	97	53	87	185	134	193	29	158
225	248	152	17	105	217	142	148	155	30	135	233	206	85	40	223
140	161	137	13	191	230	66	104	65	153	45	15	176	84	187	22

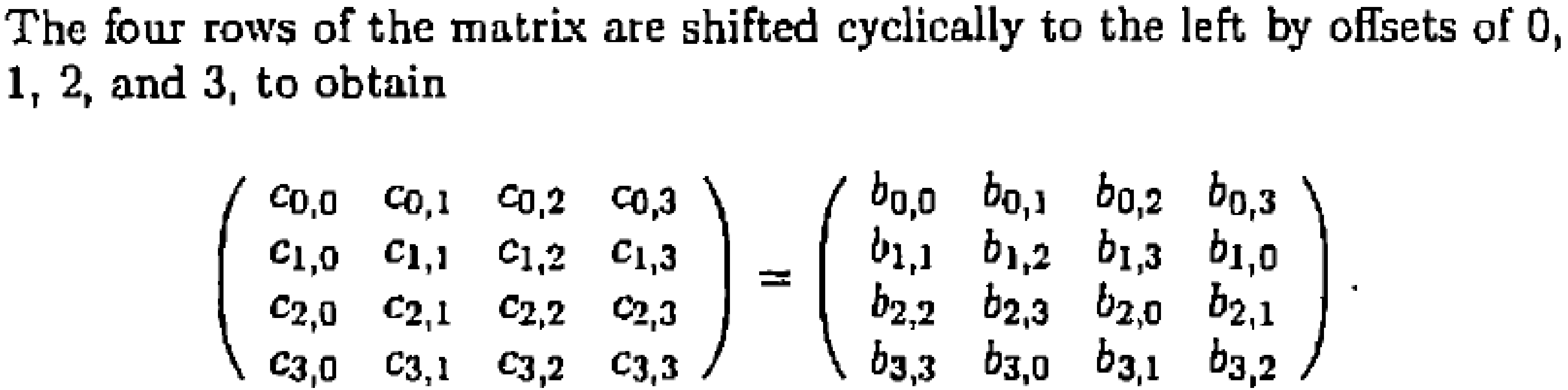
ByteSub



The output of ByteSub is again a 4 × 4 matrix of bytes, let's call it

$$\begin{pmatrix} b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} \\ b_{1,0} & b_{1,1} & b_{1,2} & b_{1,3} \\ b_{2,0} & b_{2,1} & b_{2,2} & b_{2,3} \\ b_{3,0} & b_{3,1} & b_{3,2} & b_{3,3} \end{pmatrix}.$$

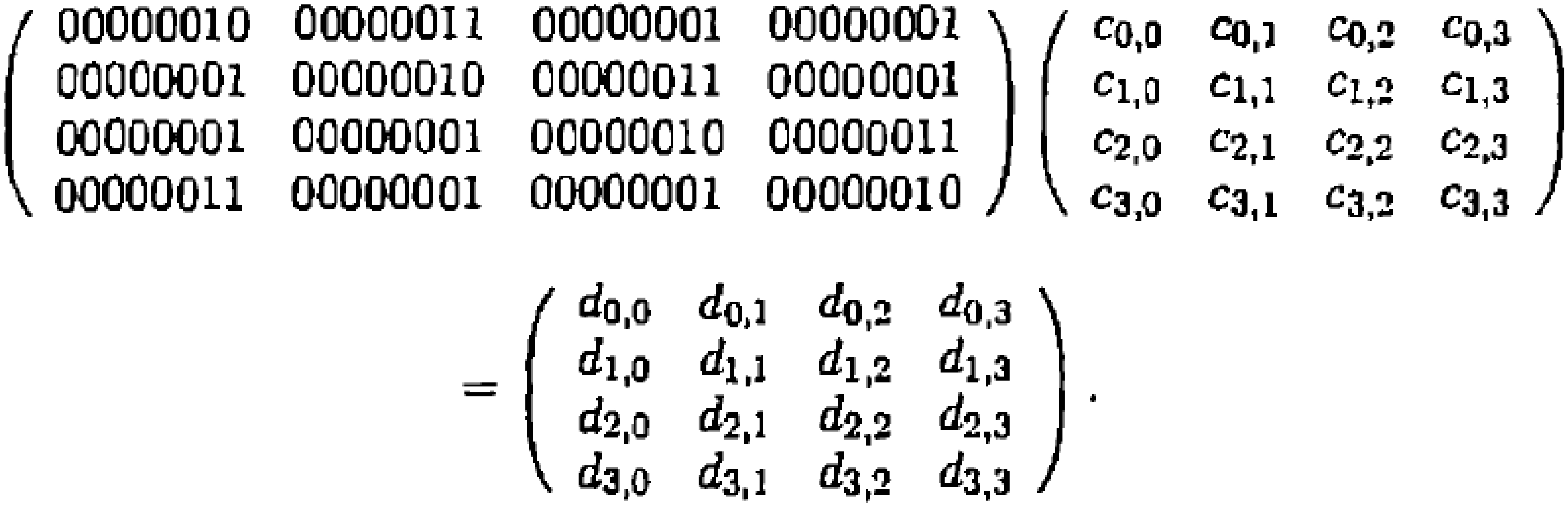
The ShiftRow Transformation



The four rows of the matrix are shifted cyclically to the left by offsets of 0, 1, 2, and 3, to obtain

$$\begin{pmatrix} c_{0,0} & c_{0,1} & c_{0,2} & c_{0,3} \\ c_{1,0} & c_{1,1} & c_{1,2} & c_{1,3} \\ c_{2,0} & c_{2,1} & c_{2,2} & c_{2,3} \\ c_{3,0} & c_{3,1} & c_{3,2} & c_{3,3} \end{pmatrix} = \begin{pmatrix} b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} \\ b_{1,1} & b_{1,2} & b_{1,3} & b_{1,0} \\ b_{2,2} & b_{2,3} & b_{2,0} & b_{2,1} \\ b_{3,3} & b_{3,0} & b_{3,1} & b_{3,2} \end{pmatrix}.$$

The MixColumn Transformation



$$\begin{pmatrix} 00000010 & 00000011 & 00000001 & 00000001 \\ 00000001 & 00000010 & 00000011 & 00000001 \\ 00000001 & 00000001 & 00000010 & 00000011 \\ 00000011 & 00000001 & 00000001 & 00000010 \end{pmatrix} \begin{pmatrix} c_{0,0} & c_{0,1} & c_{0,2} & c_{0,3} \\ c_{1,0} & c_{1,1} & c_{1,2} & c_{1,3} \\ c_{2,0} & c_{2,1} & c_{2,2} & c_{2,3} \\ c_{3,0} & c_{3,1} & c_{3,2} & c_{3,3} \end{pmatrix}$$

$$= \begin{pmatrix} d_{0,0} & d_{0,1} & d_{0,2} & d_{0,3} \\ d_{1,0} & d_{1,1} & d_{1,2} & d_{1,3} \\ d_{2,0} & d_{2,1} & d_{2,2} & d_{2,3} \\ d_{3,0} & d_{3,1} & d_{3,2} & d_{3,3} \end{pmatrix}.$$

The RoundKey Addition

- 4x4 matrix key (128 bits), $K_{i,j}$, XORed with the o/p of MixColumn step

$$
\begin{pmatrix} d_{0,0} & d_{0,1} & d_{0,2} & d_{0,3} \\ d_{1,0} & d_{1,1} & d_{1,2} & d_{1,3} \\ d_{2,0} & d_{2,1} & d_{2,2} & d_{2,3} \\ d_{3,0} & d_{3,1} & d_{3,2} & d_{3,3} \end{pmatrix} \bigoplus \begin{pmatrix} k_{0,0} & k_{0,1} & k_{0,2} & k_{0,3} \\ k_{1,0} & k_{1,1} & k_{1,2} & k_{1,3} \\ k_{2,0} & k_{2,1} & k_{2,2} & k_{2,3} \\ k_{3,0} & k_{3,1} & k_{3,2} & k_{3,3} \end{pmatrix}
$$

$$
= \begin{pmatrix} e_{0,0} & e_{0,1} & e_{0,2} & e_{0,3} \\ e_{1,0} & e_{1,1} & e_{1,2} & e_{1,3} \\ e_{2,0} & e_{2,1} & e_{2,2} & e_{2,3} \\ e_{3,0} & e_{3,1} & e_{3,2} & e_{3,3} \end{pmatrix} .
$$

The Key Schedule

- Original Key consists of 128 bits, arranged into 4x4 matrix of bytes
- Expanded by adjoining 40 more elements
- Label the first four columns *W*(0), *W*(1), *W*(2), *W*(3)
- The new columns are generated recursively
- Columns up through W(i-1) have been defined
- If *i* is not a multiple of 4, then

$$W(i) = W(i-4) \oplus W(i-1).$$

- If *i* is a multiple of 4, then

$$W(i) = W(i-4) \oplus T(W(i-1)),$$

The Key Schedule

- Where T(W(i-1)) is the transformation of W(i-1) obtained as follows:
- Let the elements of the column W(i-1) be *a, b, c, d*
- Shift these cyclically to obtain *b, c, d, a*
- Replace each of these bytes with the corresponding element in the S-box from the ByteSub step to get 4 bytes *e, f, g, h*
- Compute the round constant

$$r(i) = 00000010^{(i-4)/4}$$