Announcements

Homework 1 is released

- Available on the course website
- Due in ***two weeks***: 10/22/19 11:59pm
- Submit through **GradeScope**
  - TA Sam gave a tutorial last Wednesday

Lecture 4

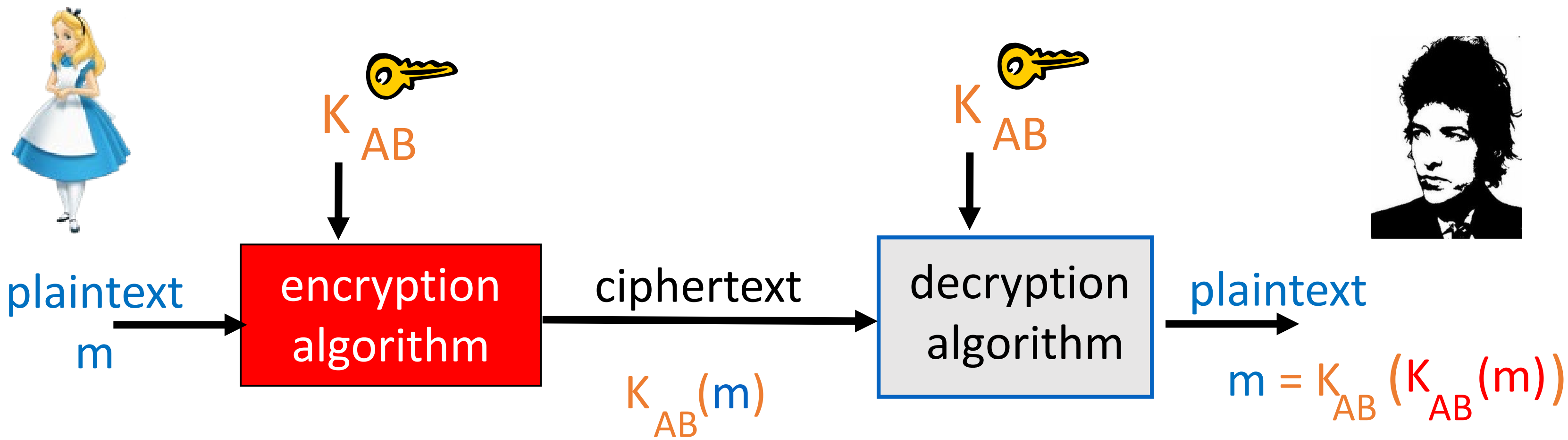
Encryption II

Suggested Readings:

- Chs 3 & 4 in KPS (recommended)
- Ch 3 in Stinson (optional)

[lecture slides are adapted from previous slides by Prof. Gene Tsudik]

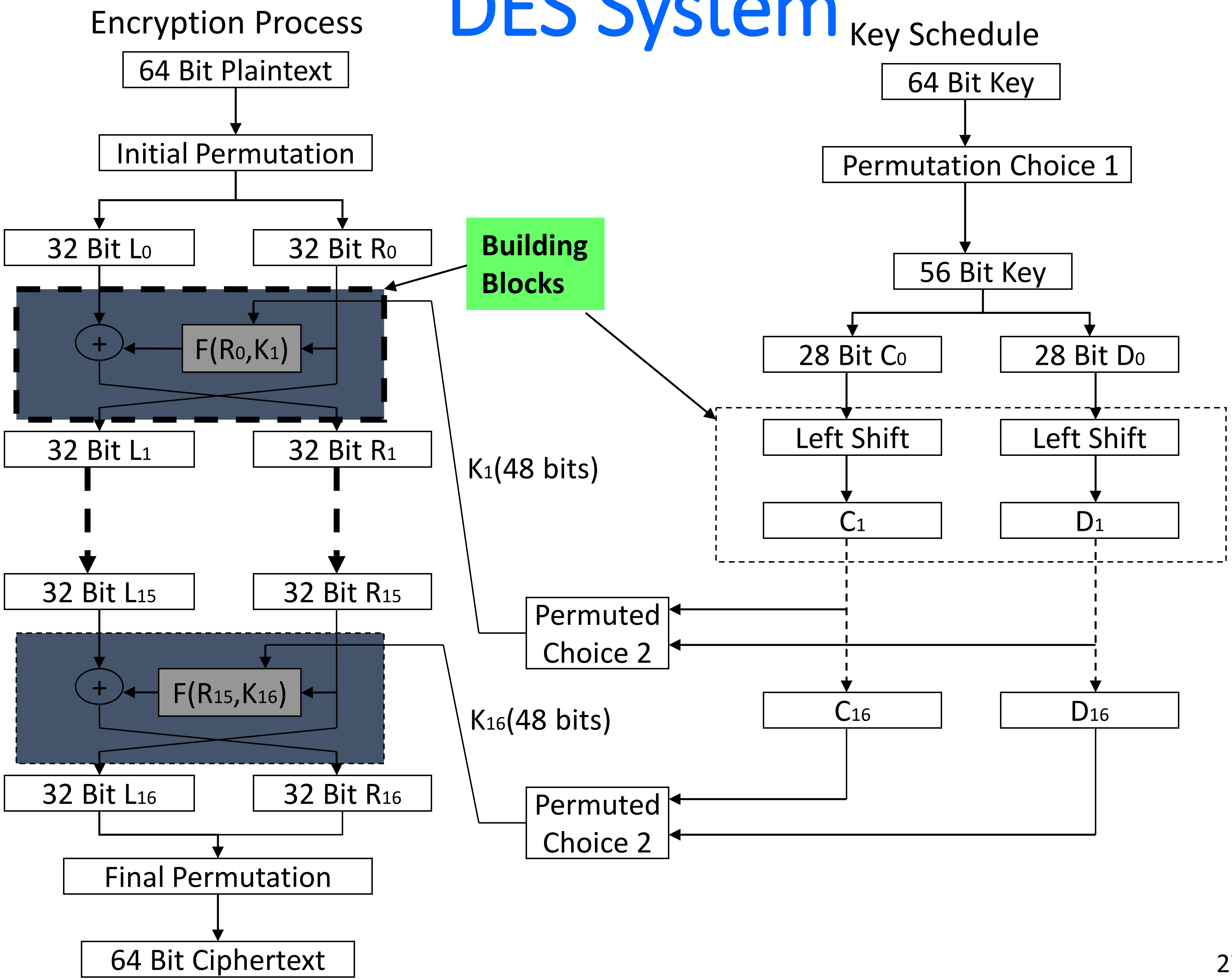
Conventional (Symmetric) Cryptography



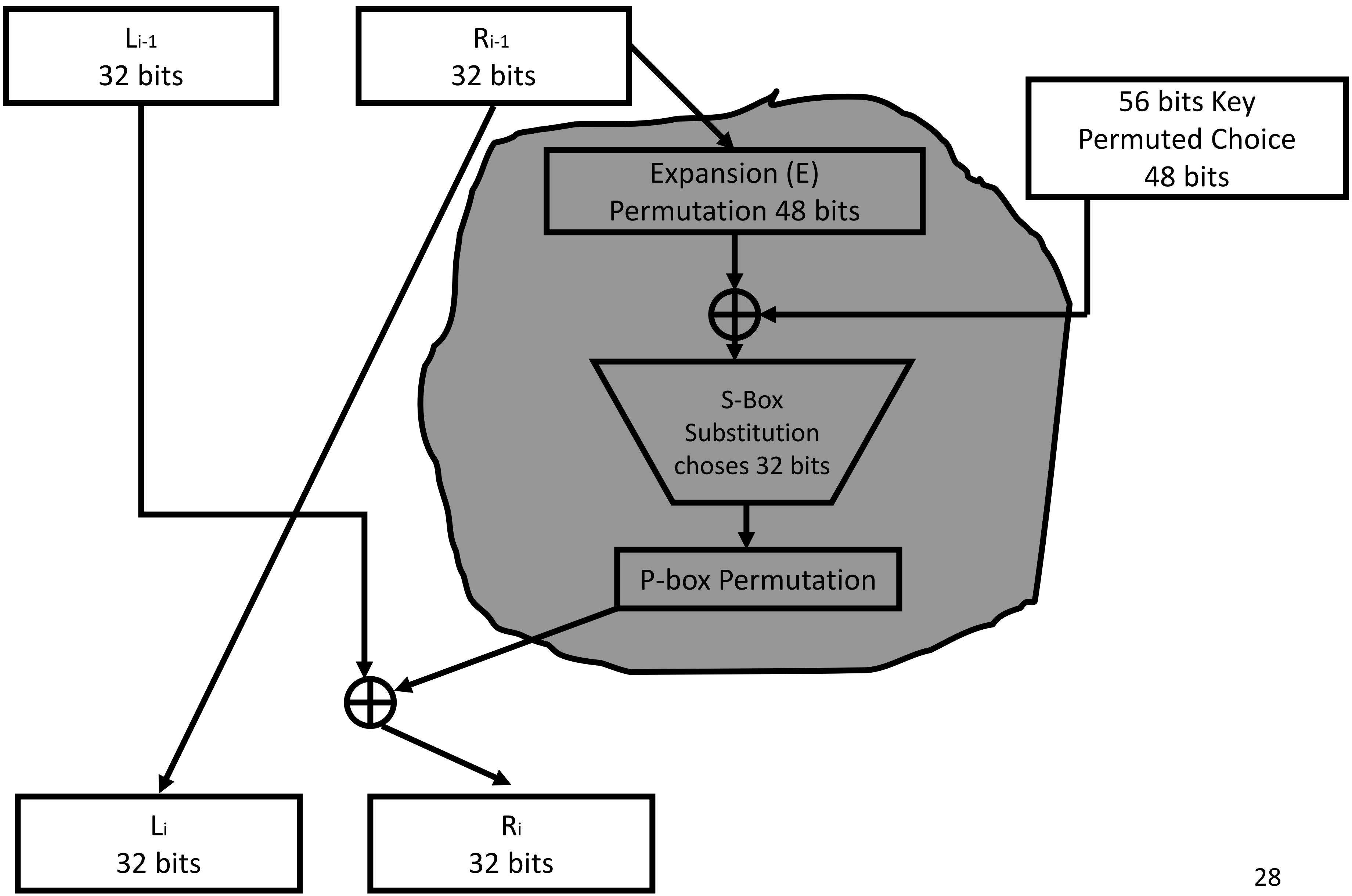
“Modern” Block Ciphers

Data Encryption Standard (DES)

DES System

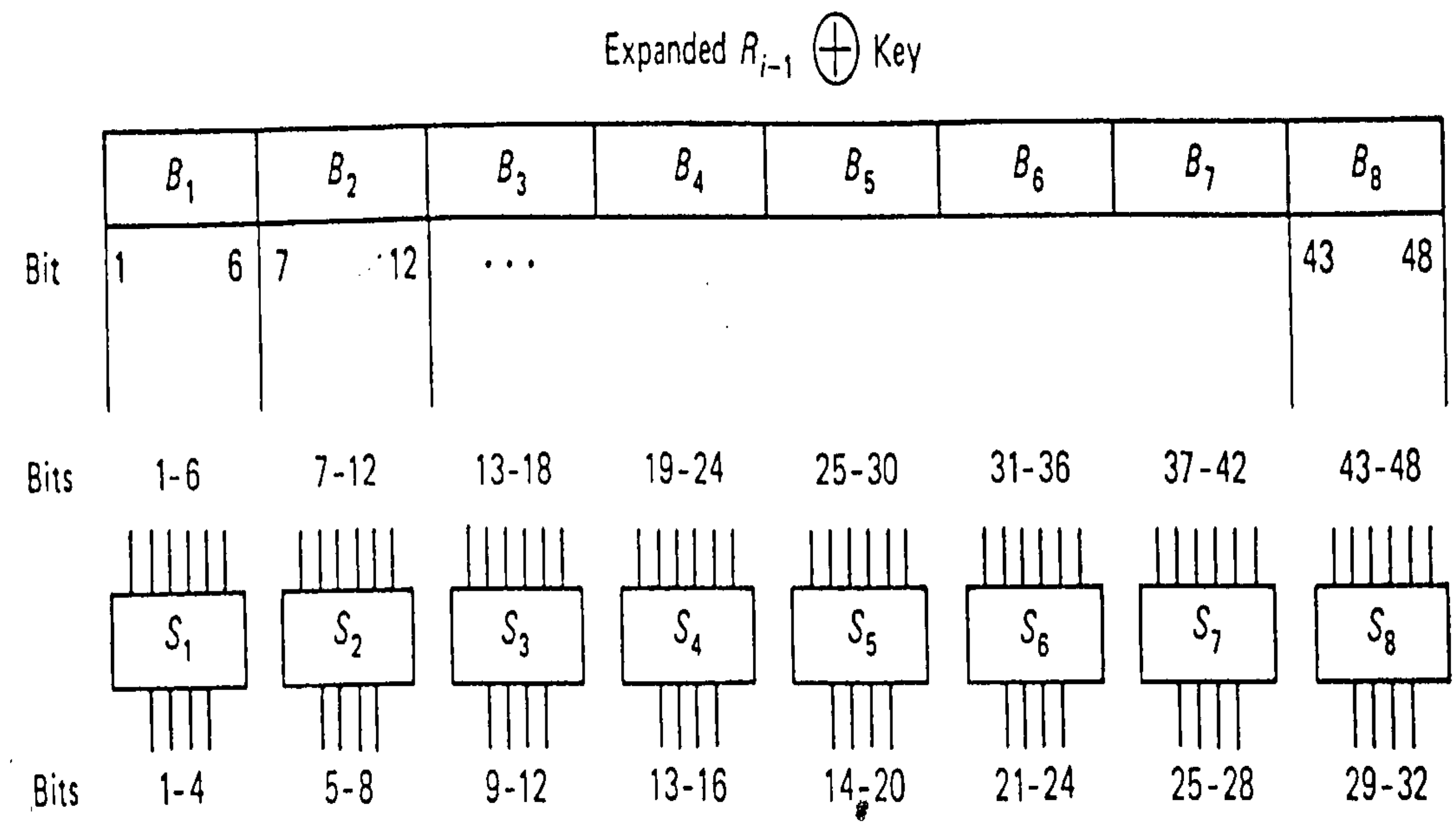

Encryption Process
64 Bit Plaintext
Initial Permutation
32 Bit L0
32 Bit R0
F(R0,K1)
32 Bit L1
32 Bit R1
32 Bit L15
32 Bit R15
F(R15,K16)
32 Bit L16
32 Bit R16
Final Permutation
64 Bit Ciphertext
Building Blocks
K1(48 bits)
K16(48 bits)
Permuted Choice 2
Permuted Choice 2
Key Schedule
64 Bit Key
Permutation Choice 1
56 Bit Key
28 Bit C0
28 Bit D0
Left Shift
Left Shift
C1
D1
C16
D16

Function F



Li-1
32 bits
Ri-1
32 bits
56 bits Key
Permuted Choice
48 bits
Expansion (E)
Permutation 48 bits
S-Box
Substitution
choses 32 bits
P-box Permutation
Li
32 bits
Ri
32 bits

DES Substitution Boxes Operation



Operation Tables of DES (IP, IP^{-1}, E and P)

Initial Pemutation (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

Bit-Selection Table E

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

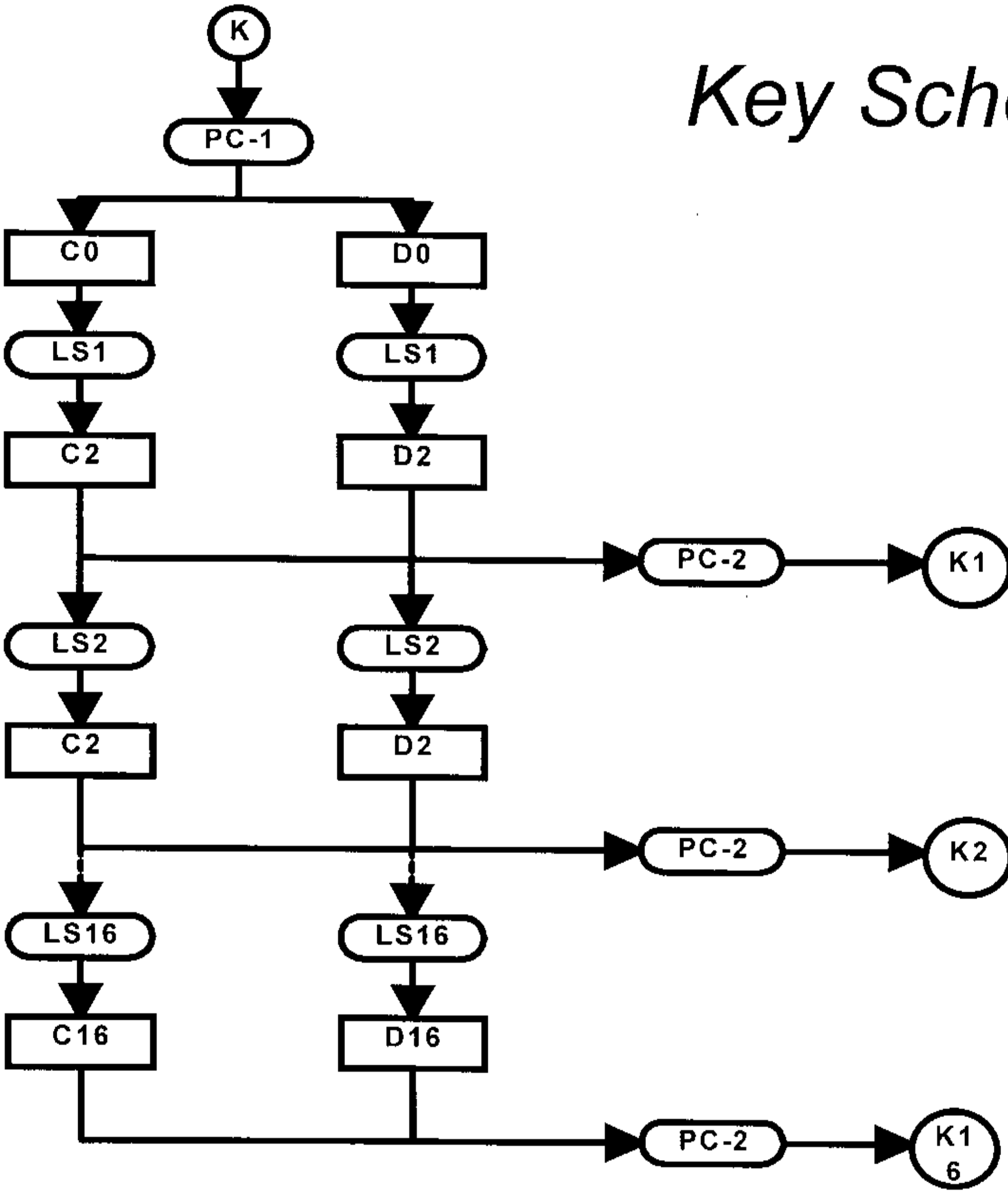
Inverse Initial Pemutation (IP^{-1})

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

Permutation P

16	7	20	21
19	12	18	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

Key Schedule -- KS



Key schedule of shifts

Iteration(i)i	No. of shifts
1	1
2	1
3	2
4	2
5	2
6	2
7	2
8	2
9	1
10	2
11	2
12	2
13	2
14	2
15	2
16	1

Key permutation PC-1

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

Key permutation PC-2

14	17	11	24	1	5
3	28	15	6	20	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	54
46	42	50	36	29	32

Operation Tables of DES

(Key Schedule,PC-1,PC-2)

Breaking DES (Cryptanalysis)

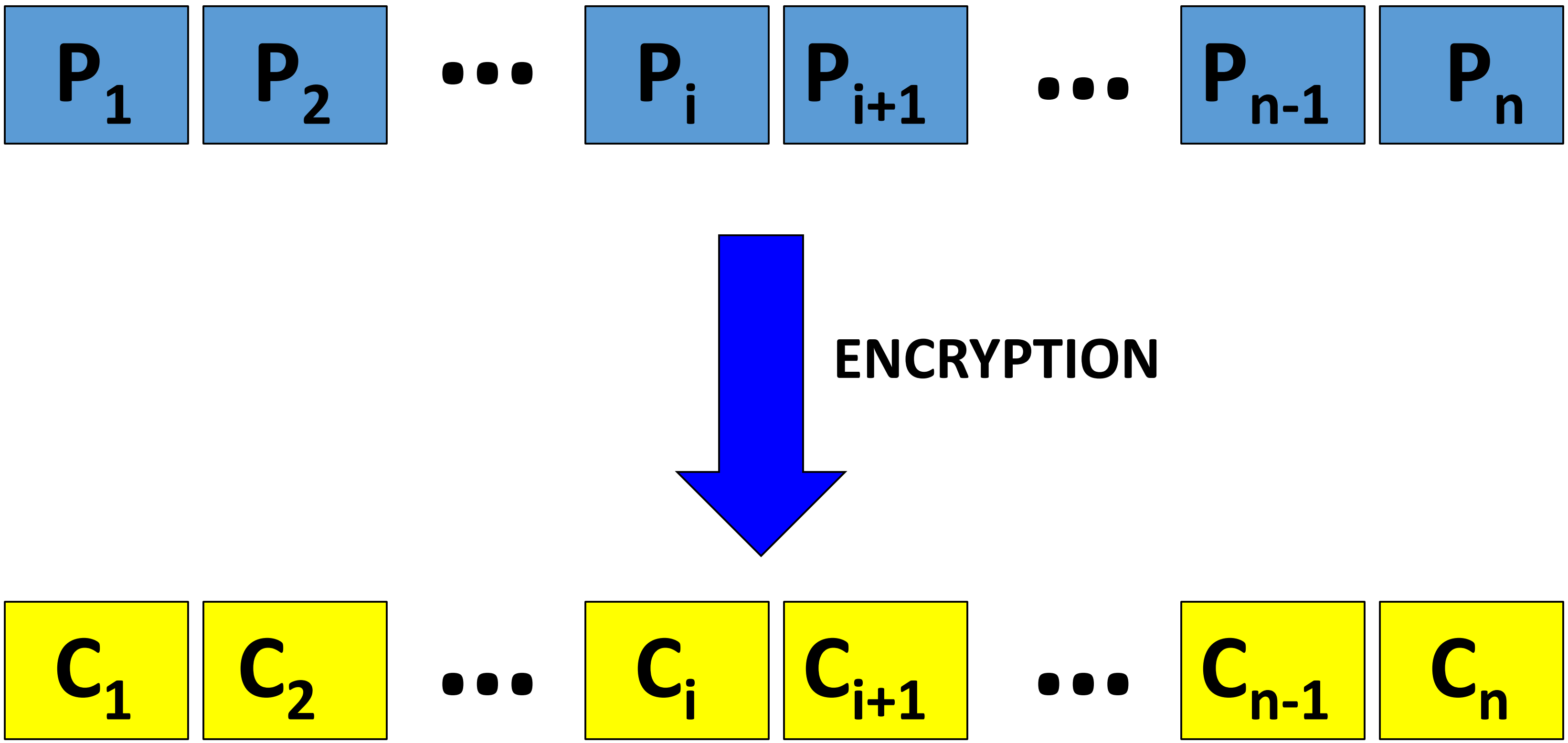
DES Key size = 56 bits

- Brute force = 2^{55} attempts on avg
- Differential cryptanalysis → 2^{47} chosen plaintexts [BS'89]
- Linear cryptanalysis → 2^{43} known plaintexts [M'93]

- More than 16 rounds do not make it any stronger
- DES Key Problems:
- Weak keys (all 0s, all 1s, a few others)
- Key size = 56 bits = 8 * 7-bit ASCII
- Alphanumeric-only password converted to uppercase

  8 * ~5-bit chars = 40 bits

Modes of Operation (not just for DES, for any block cipher)



http://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

"Native" ECB Mode

Electronic Code-Book (ECB) Mode

- Input to encryption algorithm is current plaintext block:

$$C_i = E(K, P_i)$$

$$P_i = D(K, C_i)$$

- Duplicate plaintext blocks (patterns) visible in ciphertext
  - What if Alice encrypts one word per plaintext block?
- Ciphertext block rearrangement is possible
  - To detect it, need explicit block numbering in plaintext
- Parallel encryption and decryption (random access)
- Error in one ciphertext block ➔ one-block loss
- One-block loss in ciphertext?

CBC Mode

Cipher-Block Chaining (CBC) Mode

- Input to encryption algorithm is the XOR of current plaintext block and preceding ciphertext block:

$$C_i = E(K, P_i \textbf{ XOR } C_{i-1}) \qquad C_0 = IV$$

$$P_i = D(K, C_i) \textbf{ XOR } C_{i-1}$$

- Duplicate plaintext blocks (patterns) NOT exposed
- Block rearrangement is detectable
- No parallel encryption
  - How about parallel decryption?
- Error in one ciphertext block → two-block loss
- One-block ciphertext loss?

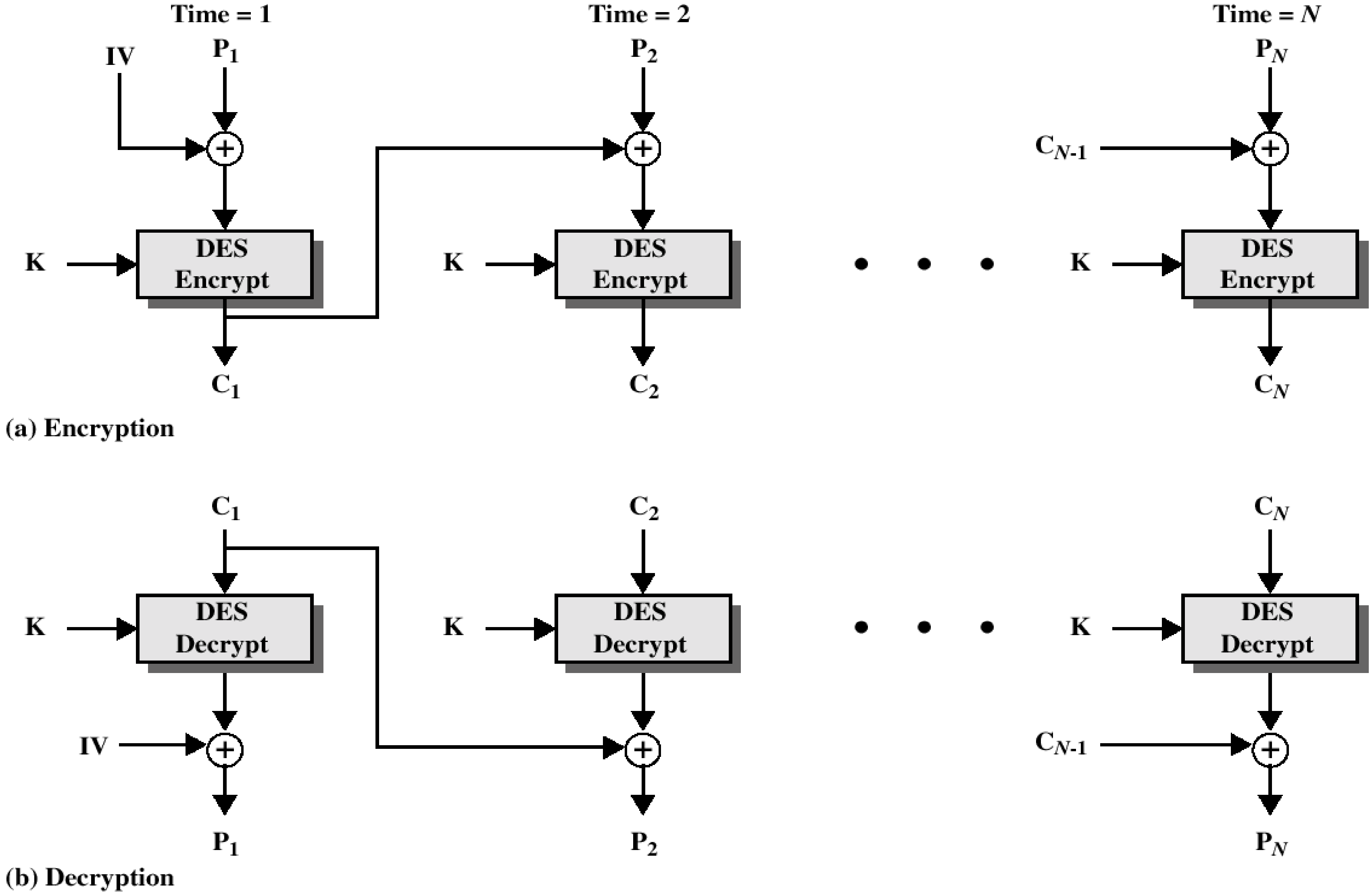


Figure 2.7 Cipher Block Chaining (CBC) Mode

OFB Mode

Output Feedback (OFB) Mode

- Key-stream is produced by repeated encryption of V_o:

$$C_i = E(K, V_{i-1})\ \textbf{XOR}\ P_i \qquad V_0 = IV, \ldots, V_i = E(K, V_{i-1})$$

$$P_i = E(K, V_{i-1})\ \textbf{XOR}\ C_i$$

- Duplicate plaintext blocks (patterns) NOT exposed
- Block rearrangement is detectable
- Key-stream is independent of plaintext
  - How does that affect speed of encryption? Parallelism?
- Bit error in one ciphertext block ➔ one-bit error in plaintext
- One-block ciphertext loss ➔ big mess ☺
- Can encrypt less than block size

CFB Mode

Cipher Feedback (CFB) Mode

- Key-stream is produced by re-encryption of preceding ciphertext -- C_{i-1}:

$$C_i = P_i \textbf{ XOR } E(K, C_{i-1}) \quad C_0 = IV$$

$$P_i = E(K, C_{i-1}) \textbf{ XOR } C_i$$

- Duplicate plaintext blocks (patterns) NOT exposed
- Block rearrangement is detectable
- Key-stream is **dependent on** plaintext
  - How does that affect speed of encryption? Parallelism?
- Bit error in one ciphertext block → one-bit + one-block loss in plaintext
  - Adversary can still selectively flip/change bits
- One-block ciphertext loss → 1-extra-block loss
- Can encrypt less than block size

CTR Mode

Counter (CTR) Mode

- Key-stream is produced by encryption increasing counter:

$$C_i = E(K, CTR) \textbf{ XOR } P_i \qquad CTR{+}{+}$$

$$P_i = E(K, CTR) \textbf{ XOR } C_i$$

- Duplicate plaintext blocks (patterns) NOT exposed, **unless**?
- Block rearrangement is detectable
- Key-stream is independent of plaintext
- Parallel encryption and decryption (random access)
- Bit error in one ciphertext block ➔ one-bit error in plaintext
- One-block ciphertext loss ➔ big mess
- Can encrypt less than block size

MAC Mode

Message Authentication Code (MAC) Mode

- Encryption is the same as in CBC mode, but, ciphertext is NOT sent!

$C_i = E(K, P_i \textbf{ XOR } C_{i-1})$ $C_0 = IV$

What is sent or stored: $P_1, \ldots, P_n$, C_n = MAC

Receiver recomputes C_n with K and compares

- Any change in plaintext results in unpredictable changes in MAC

How to strengthen DES: the case of double DES

- **2DES: C = DES (K1, DES (K2, P))**
- **Seems to be hard to break by "brute force", approx. 2^{111} trials**
- **Assume Eve is trying to break 2DES and has a single (P,C) pair**

Meet-in-the-middle ATTACK:

I. For each possible K'_i (where $0 < i < 2^{56}$)
   1. Compute $C'_i = DES(K'_i, P)$
   2. Store: $[C'_i, K'_i]$ in look-up table T (indexed by C'_i)

II. For each possible K''_i (where $0 < i < 2^{56}$)
   1. Compute $C''_i = DES^{-1}(K''_i, C)$
   2. Look up C''_i in T
   3. If lookup succeeds, output: $K1=K'_i$, $K2=K''_i$

TOTAL COST: $O(2^{56}+2^{56})$ operations + $O(2^{64})$ storage

DES Variants

- **2-DES:**
  - **C = E(K2,E(K1, P)) → 57 effective key bits (meet-in-the-middle attack)**
- **3-DES (Triple DES)**
  - **C = E(K3, D(K2, E(K1,P))) → 112 effective key bits (meet-in-the-middle attack)**
  - **C = E(K1, D(K2, E(K1,P))) → <=80 effective key bits**
- **DESX**
  - **C= K3 XOR E(K2, (K1 XOR P)) → seems like 184 key bits**
  - **Effective key bits → approx. 118**
- **Another simple variation:**
  - **C = K2 XOR E(K1, P) → weak!**

NOTE: The same variants can be constructed out of any cipher

DES Variants

Why does 3-DES (or generally n-DES) work?

Because, as a function, DES is not a group…

A "group" is an algebraic structure. One of its properties is that, taking any 2 elements of the group (a,b) and applying an operator F() yields another element c in the group.

Suppose: C = DES(K1,DES(K2,P))

There is no K, such that:

for each possible plaintext P, DES(K,P) = C

DES Summary

- Feistel network based block cipher
- 64-bit data blocks
- 56-bit keys (8 parity bits)
- 16 rounds (shifts, XORs)
- Key schedule
- S-box selection secret …

- DES “aging”
- 2-DES: meet-in-the-middle attack
- 3-DES: 112-bit security
- DESX: 118-bit security

Advanced Encryption Standard (AES): The Rijndael Block Cipher

Introduction and History

- National Institute of Science and Technology (NIST) regulates standardization in the US
- By mid-90s, DES was an aging standard that no longer met the needs for strong commercial-grade encryption
- Triple-DES: Endorsed by NIST as a “de facto” standard
- But … slow in software and large footprint (code size)
- **Advanced Encryption Standard (AES)**
  - Goal is to define the Federal Information Processing Standard (FIPS) by selecting a new encryption algorithm suitable for encrypting (non-classified non-military) government documents
  - Candidate algorithms must be:
    - Symmetric-key ciphers supporting 128, 192, and 256 bit keys
    - Royalty-Free
    - Unclassified (i.e., public domain)
    - Available for worldwide export
  - 1997: NIST publishes request for proposal
  - 1998-1999: 15 submissions -> 5 finalists
  - 2000: NIST chooses Rijndael as AES

Introduction and History

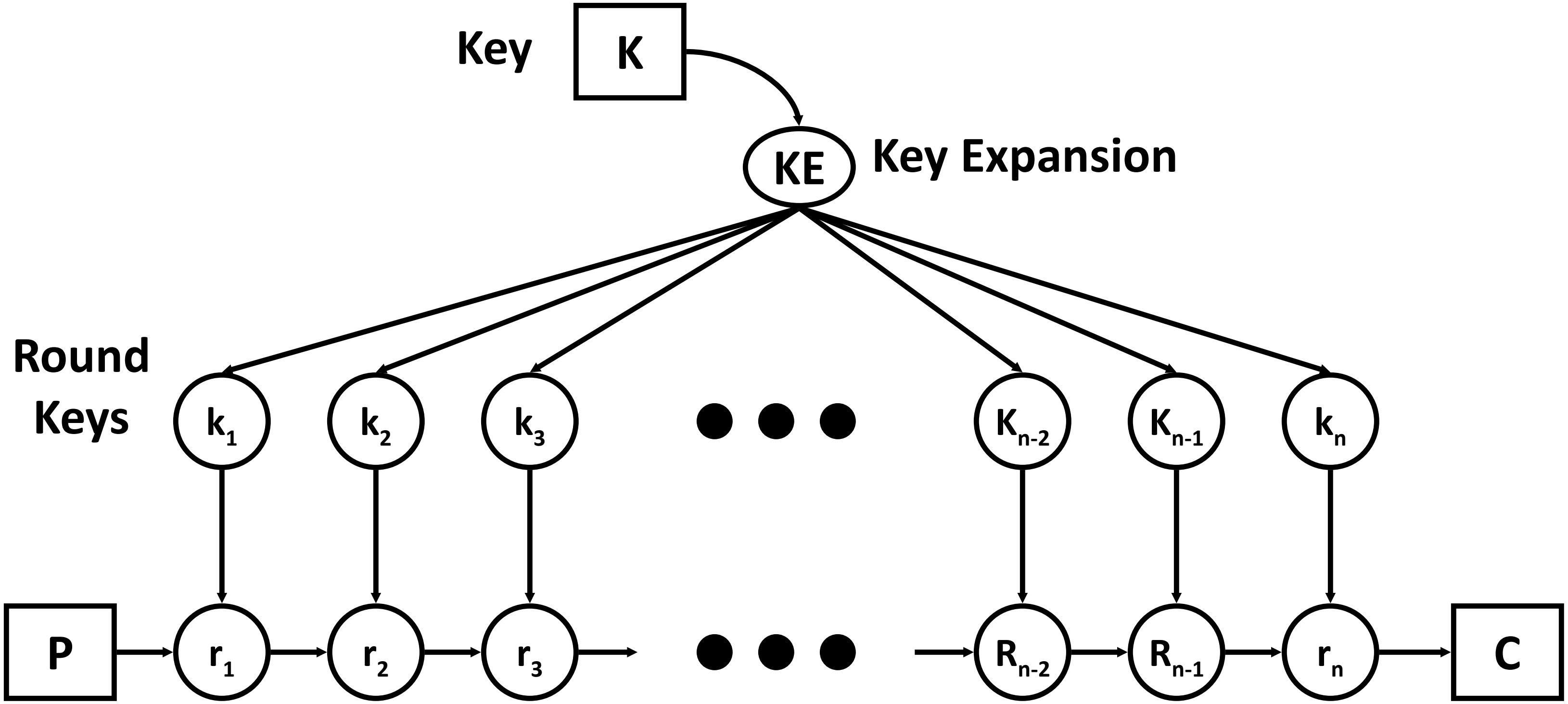
- AES Round-3 Finalist Algorithms (ranked by vote # in AES Round-2, high to low):
  - Rijndael
    - by Joan Daemen and Vincent Rijmen (Belgium)
  - Serpent
    - by Ross Anderson (UK), Eli Biham (ISR) and Lars Knudsen (NO)
  - Twofish
    - From Counterpane Internet Security, Inc. (MN)
  - RC6
    - By Ron Rivest of MIT & RSA Labs, creator of the widely used RC4/RC5 algorithm and “R” in RSA
  - MARS
    - Candidate offering from IBM Research

Rijndael

The Winner: Rijndael

- Joan Daemen (of Proton World International) and Vincent Rijmen (of Katholieke Universiteit Leuven).
- Pronounced "Rhine-doll"
- Allows only 128, 192, and 256-bit key sizes (unlike other candidates)
- Variable input block length: 128, 192, or 256 bits. All nine combinations of key-block length possible.
  - A block is the smallest data size the algorithm will encrypt
- Vast speed improvement over DES in both hw and sw implementations
  - 8,416 bytes/sec on a 20MHz 8051
  - 8.8 Mbytes/sec on a 200MHz Pentium Pro

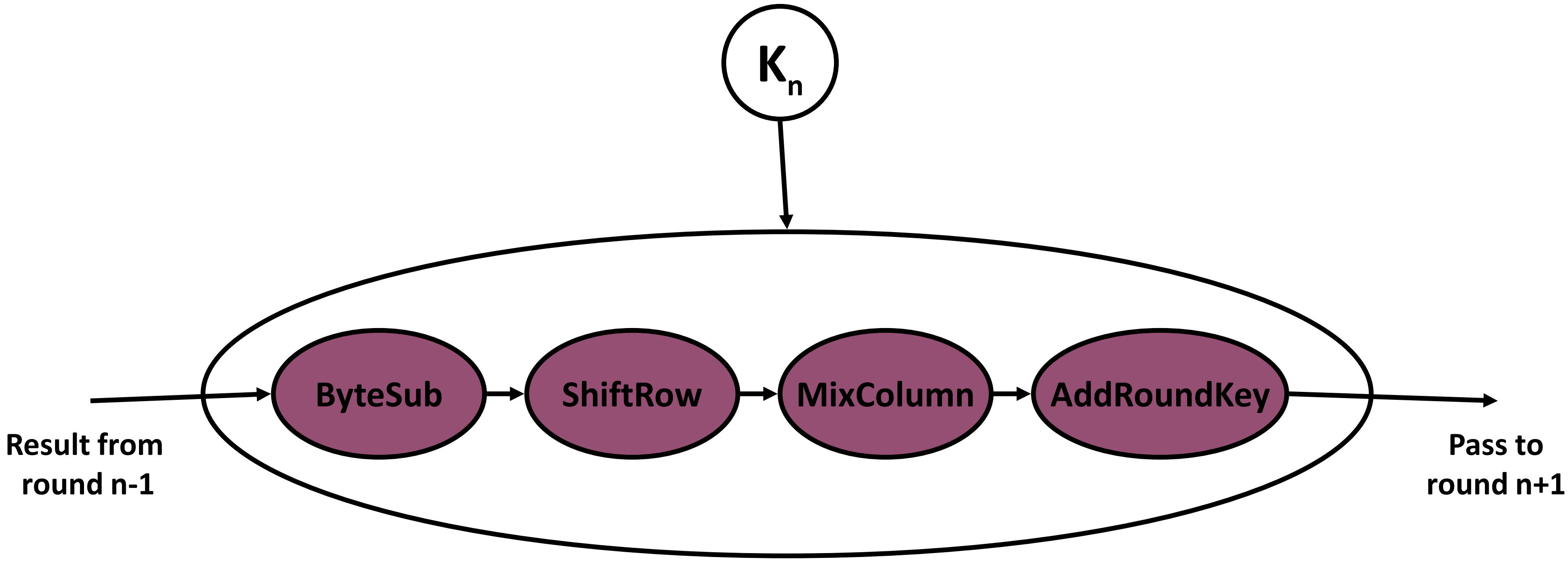
Rijndael



Encryption Rounds r_1 … r_n

- Key is expanded to a set of n round keys
- Input block P put thru n rounds, each with a distinct round sub-key.
- Strength of algorithm relies on difficulty of obtaining intermediate results (or *state*) of round i from round i+1 without the round key.

Rijndael



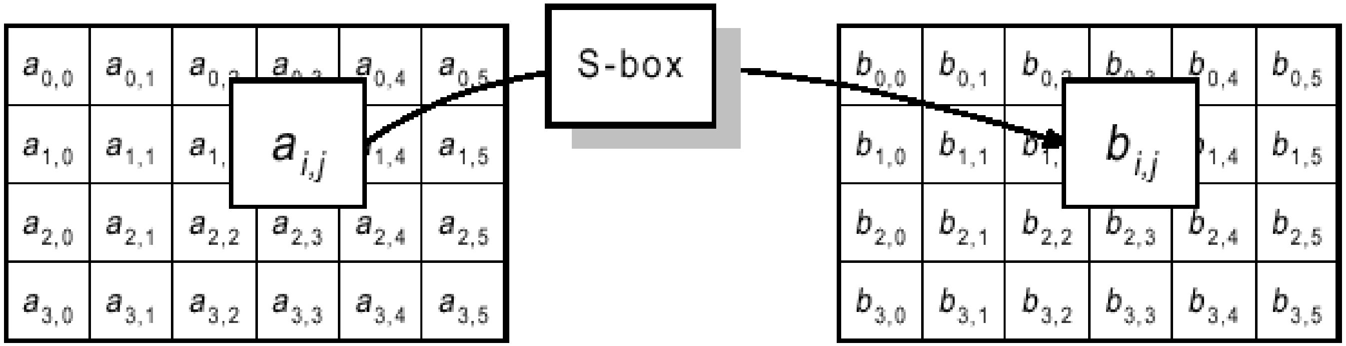
Detailed view of round n

- **Each round performs the following operations:**
  - **Non-linear Layer:** No linear relationship between the input and output of a round
  - **Linear Mixing Layer:** Guarantees high diffusion over multiple rounds
    - Very small correlation between bytes of the round input and the bytes of the output
  - **Key Addition Layer:** Bytes of the input are simply XOR'ed with the expanded round key

Rijndael

- Three layers provide strength against known types of cryptographic attacks: Rijndael provides "full diffusion" after only two rounds
- Cryptanalysis
  - Key recovery attack:
    - Best one only 4x faster than exhaustive search [BKR'11]
  - Related key attack:
    - AES-256: Given 2^99 input/output pairs from 4 related keys in AES-256 can recover keys in time 2^99 [BK'09]
    - However, how realistic is that?

Rijndael: ByteSub



Each byte at the input of a round undergoes a non-linear byte substitution according to the following transform:

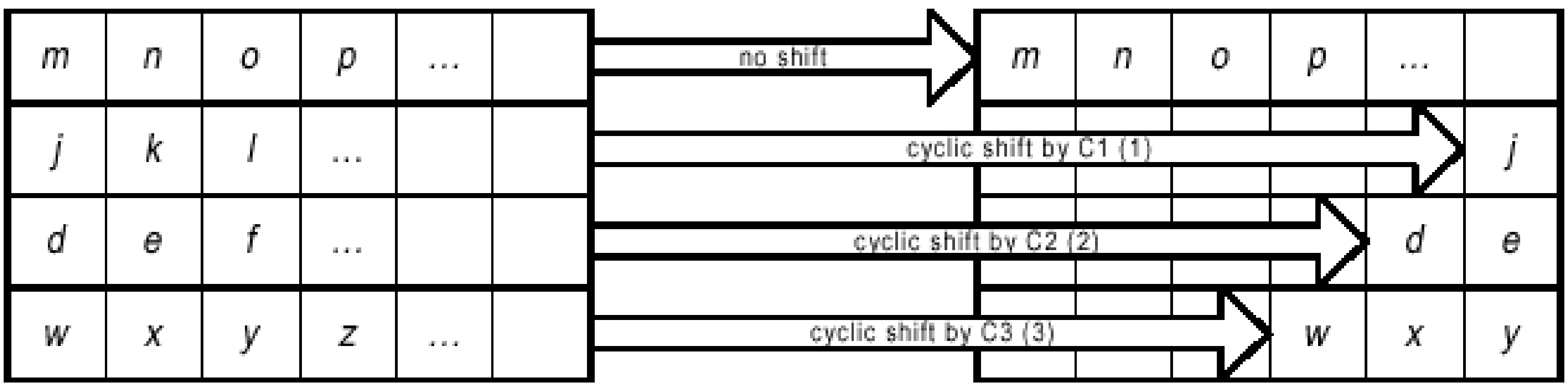
$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Substitution ("S")-box

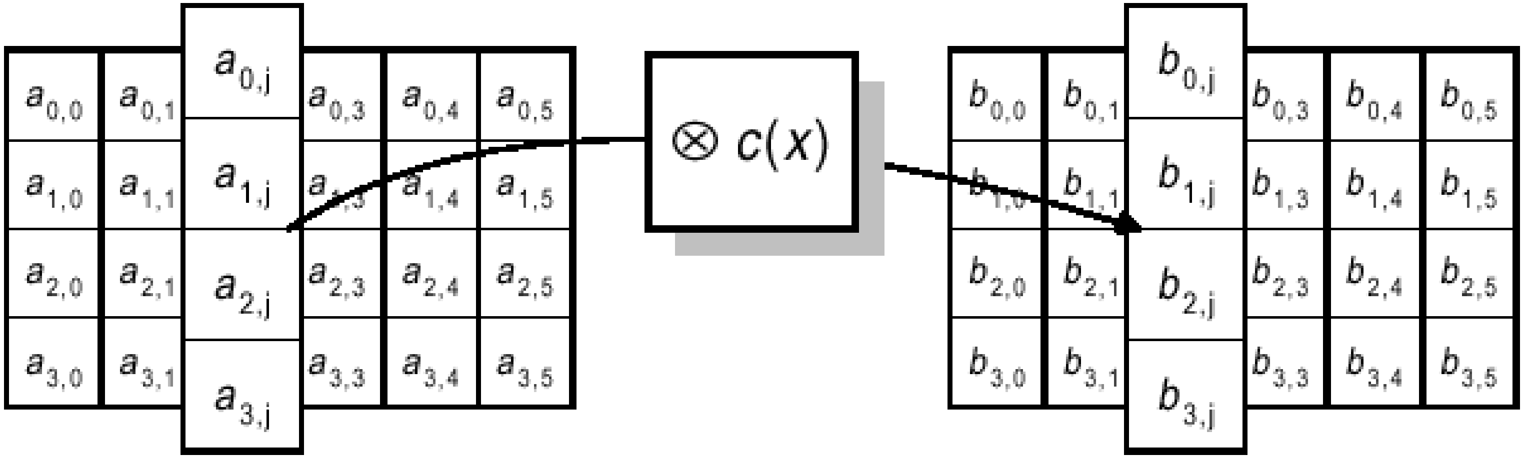
Rijndael: ShiftRow

Nb	C1	C2	C3
4	1	2	3
6	1	2	3
8	1	3	4

Depending on the block length, each “row” of the block is cyclically shifted according to the above table



Rijndael: MixColumn



Each column is multiplied by a fixed polynomial

C(x) = ’03’*X^3 + ’01’*X^2 + ’01’*X + ’02’

This corresponds to matrix multiplication b(x) = c(x) ⊗ a(x):

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

Not XOR

Rijndael: Key Expansion and Addition

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$	$a_{0,4}$	$a_{0,5}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,4}$	$a_{1,5}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	$a_{2,4}$	$a_{2,5}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$	$a_{3,4}$	$a_{3,5}$

$\oplus$

$k_{0,0}$	$k_{0,1}$	$k_{0,2}$	$k_{0,3}$	$k_{0,4}$	$k_{0,5}$
$k_{1,0}$	$k_{1,1}$	$k_{1,2}$	$k_{1,3}$	$k_{1,4}$	$k_{1,5}$
$k_{2,0}$	$k_{2,1}$	$k_{2,2}$	$k_{2,3}$	$k_{2,4}$	$k_{2,5}$
$k_{3,0}$	$k_{3,1}$	$k_{3,2}$	$k_{3,3}$	$k_{3,4}$	$k_{3,5}$

=

$b_{0,0}$	$b_{0,1}$	$b_{0,2}$	$b_{0,3}$	$b_{0,4}$	$b_{0,5}$
$b_{1,0}$	$b_{1,1}$	$b_{1,2}$	$b_{1,3}$	$b_{1,4}$	$b_{1,5}$
$b_{2,0}$	$b_{2,1}$	$b_{2,2}$	$b_{2,3}$	$b_{2,4}$	$b_{2,5}$
$b_{3,0}$	$b_{3,1}$	$b_{3,2}$	$b_{3,3}$	$b_{3,4}$	$b_{3,5}$

Each word is simply XOR'ed with the expanded round key

Key Expansion algorithm:

```
KeyExpansion(int* Key[4*Nk], int* EKey[Nb*(Nr+1)])
{
    for(i = 0; i < Nk; i++)
        EKey[i] = (Key[4*i],Key[4*i+1],Key[4*i+2],Key[4*i+3]);
    for(i = Nk; i < Nb * (Nr + 1); i++)
    {
        temp = EKey[i - 1];
        if (i % Nk == 0)
            temp = SubByte(RotByte(temp)) ^ Rcon[i / Nk];
        EKey[i] = EKey[i - Nk] ^ temp;
    }
}
```