Cryptography and Network Security (CS435/890BN)

Part Three

(Modern Symmetric Ciphers)

Block vs Stream Ciphers

- block ciphers process messages in blocks, each of which is then en/decrypted
- like a substitution on very big characters
  - 64-bits or more
- stream ciphers process messages a bit or byte at a time when en/decrypting
- many current ciphers are block ciphers
- broader range of applications

Stream Ciphers

- process message bit by bit (as a stream)
- have a pseudo random **keystream**
- combined (XOR) with plaintext bit by bit
- randomness of **stream key** completely destroys statistically properties in message
  - $C_i = M_i$ XOR $StreamKey_i$
- but must never reuse stream key
  - otherwise can recover messages

Stream Cipher

Encrypts a digital data stream one bit or one byte at a time

Examples:

- Autokeyed Vigenère cipher
- Vernam cipher

In the ideal case, a one-time pad version of the Vernam cipher would be used, in which the keystream is as long as the plaintext bit stream

If the cryptographic keystream is random, then this cipher is unbreakable by any means other than acquiring the keystream

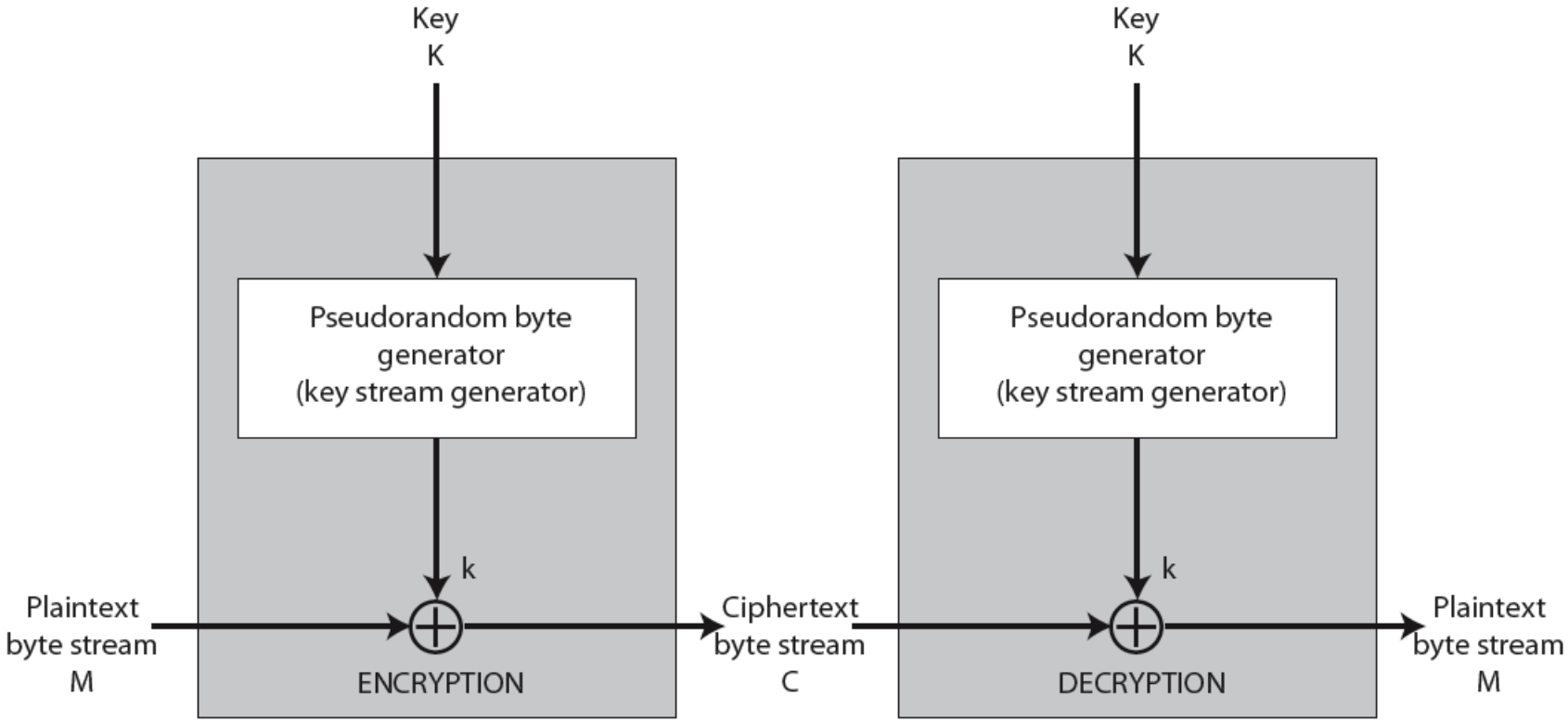
- Keystream must be provided to both users in advance via some independent and secure channel
- This introduces insurmountable logistical problems if the intended data traffic is very large

For practical reasons the bit-stream generator must be implemented as an algorithmic procedure so that the cryptographic bit stream can be produced by both users

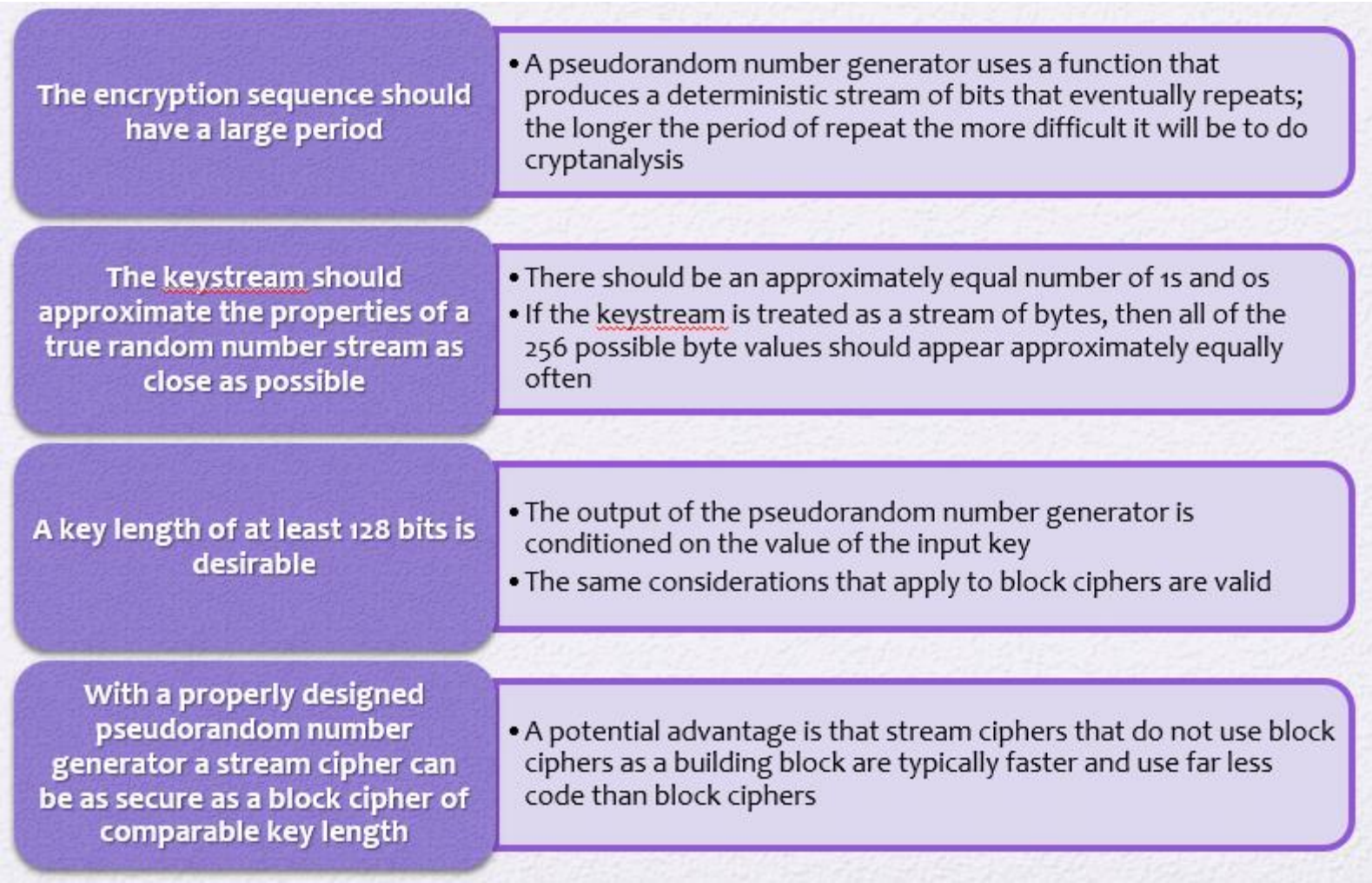
It must be computationally impractical to predict future portions of the bit stream based on previous portions of the bit stream

The two users need only share the generating key and each can produce the keystream

Stream Cipher Structure



Stream Cipher Design Considerations



- Properly designed, can be as secure as a block cipher with same size key

RC4

- Designed in 1987 by Ron Rivest for RSA Security
- Variable key size stream cipher with byte-oriented operations
- Based on the use of a random permutation
- Eight to sixteen machine operations are required per output byte and the cipher can be expected to run very quickly in software
- Used in the Secure Sockets Layer/Transport Layer Security (SSL/TLS) standards that have been defined for communication between Web browsers and servers
- Is also used in the Wired Equivalent Privacy (WEP) protocol and the newer WiFi Protected Access (WPA) protocol that are part of the IEEE 802.11 wireless LAN standard

RC4 Key Schedule

- starts with an array S of numbers: 0..255
- use key to well and truly shuffle
- S forms **internal state** of the cipher

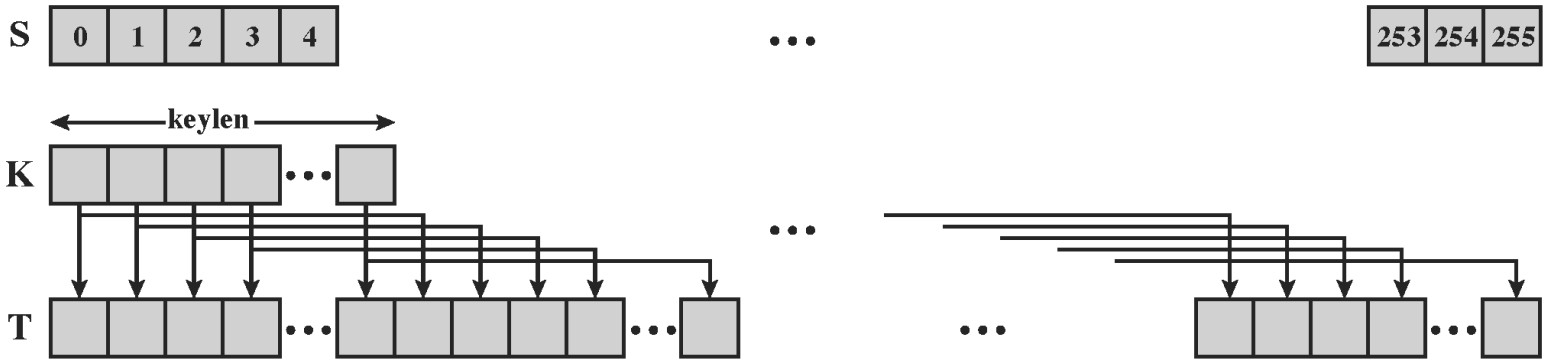
```
for i = 0 to 255 do
   S[i] = i
   T[i] = K[i mod keylen])
j = 0
for i = 0 to 255 do
   j = (j + S[i] + T[i]) (mod 256)
   swap (S[i], S[j])
```

RC4 Encryption

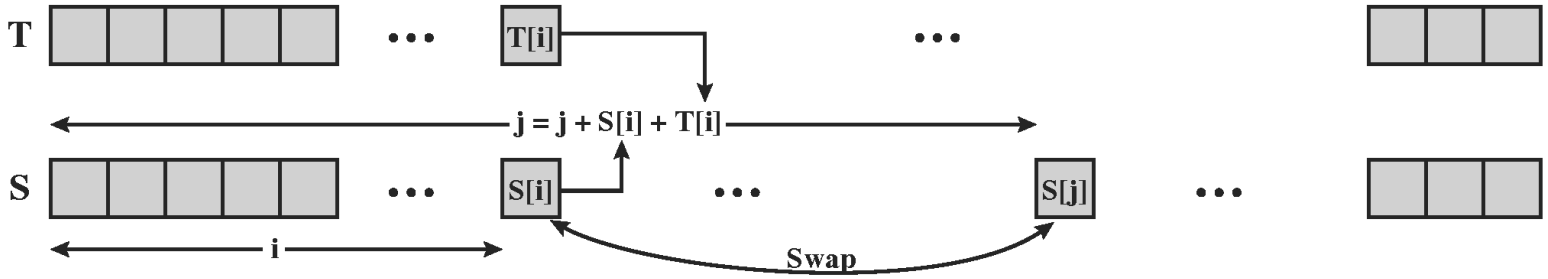
- encryption continues shuffling array values
- sum of shuffled pair selects "stream key" value from permutation
- XOR S[t] with next byte of message to en/decrypt

```
i = j = 0
for each message byte M_i
    i = (i + 1) (mod 256)
    j = (j + S[i]) (mod 256)
    swap(S[i], S[j])
    t = (S[i] + S[j]) (mod 256)
    C_i = M_i XOR S[t]
```

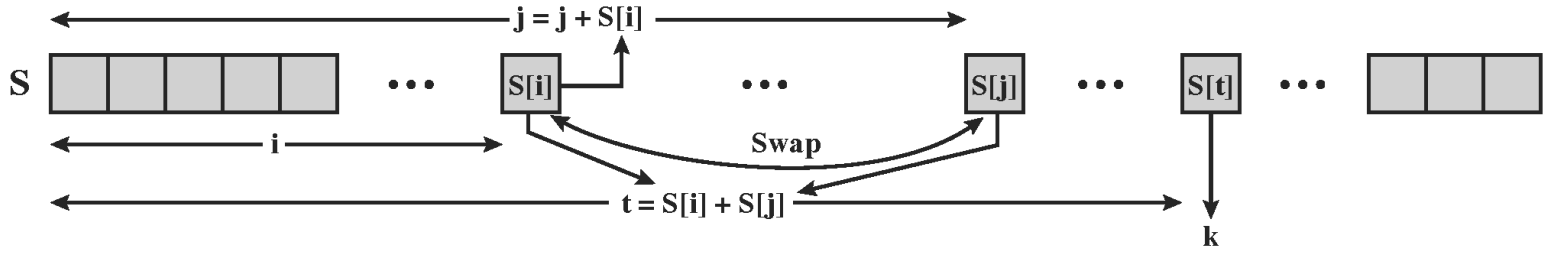
RC4 Overview



(a) Initial state of S and T



(b) Initial permutation of S



(c) Stream Generation

RC4 Security

A number of papers have been published analyzing methods of attacking RC4

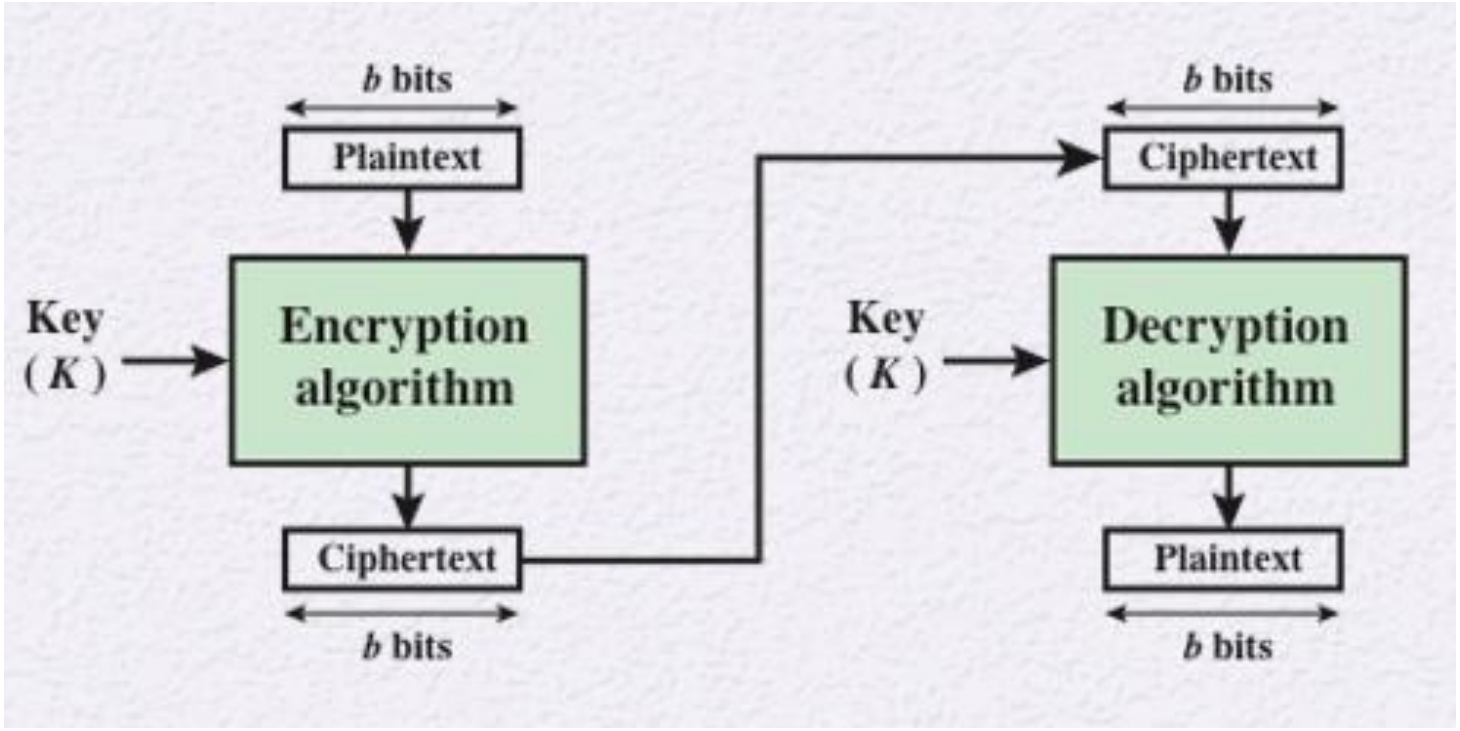
- None of these approaches is practical against RC4 with a reasonable key length

A more serious problem is that the WEP protocol intended to provide confidentiality on 802.11 wireless LAN networks is vulnerable to a particular attack approach

- The problem is not with RC4 itself, but the way in which keys are generated for use as input
- Problem does not appear to be relevant to other applications and can be remedied in WEP by changing the way in which keys are generated
- Problem points out the difficulty in designing a secure system that involves both cryptographic functions and protocols that make use of them

- since RC4 is a stream cipher, must **never reuse a key**

Block Cipher



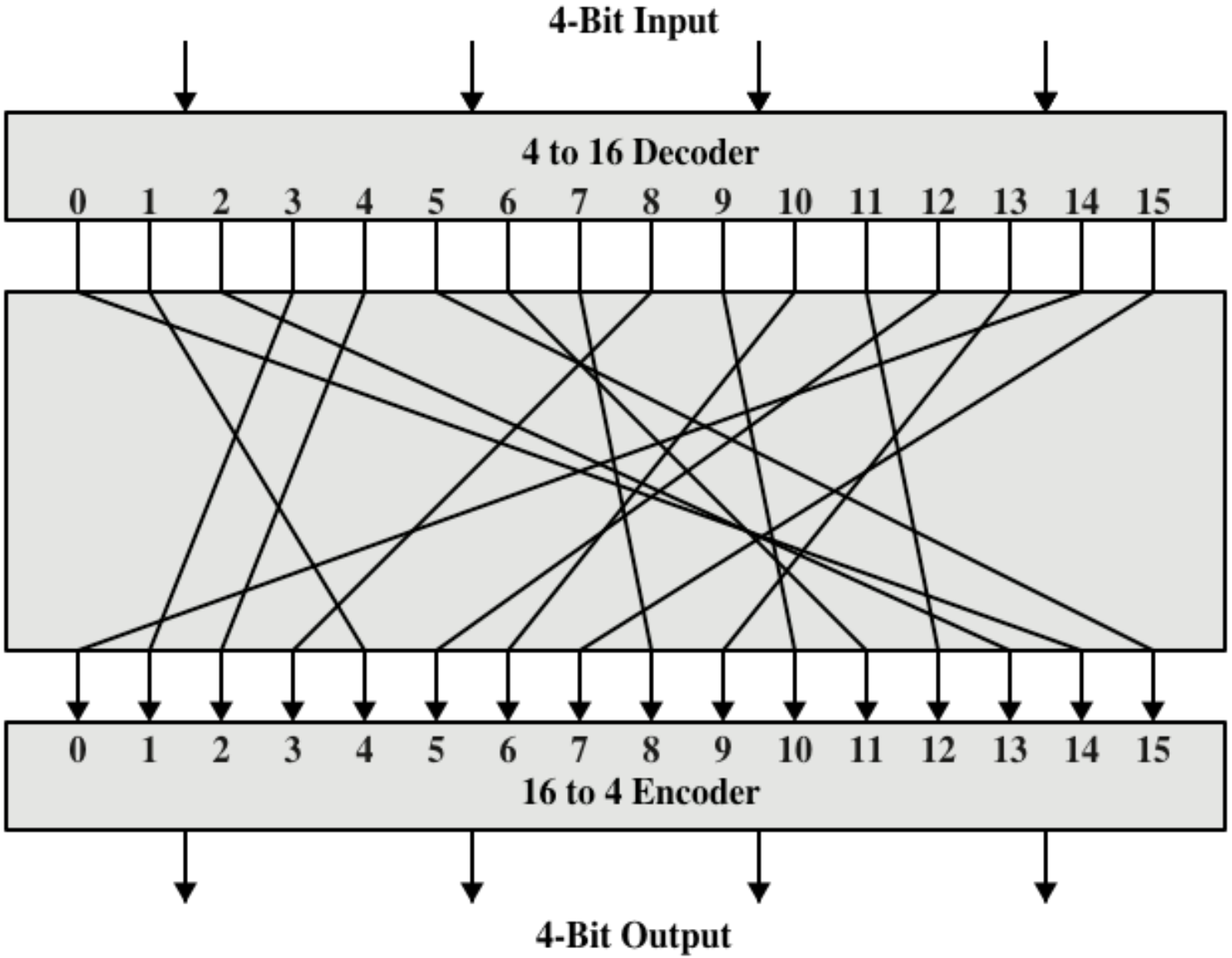


Figure 4.2 General *n*-bit-*n*-bit Block Substitution (shown with $n = 4$)

Encryption and Decryption Tables for Substitution Cipher of Figure 4.2

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

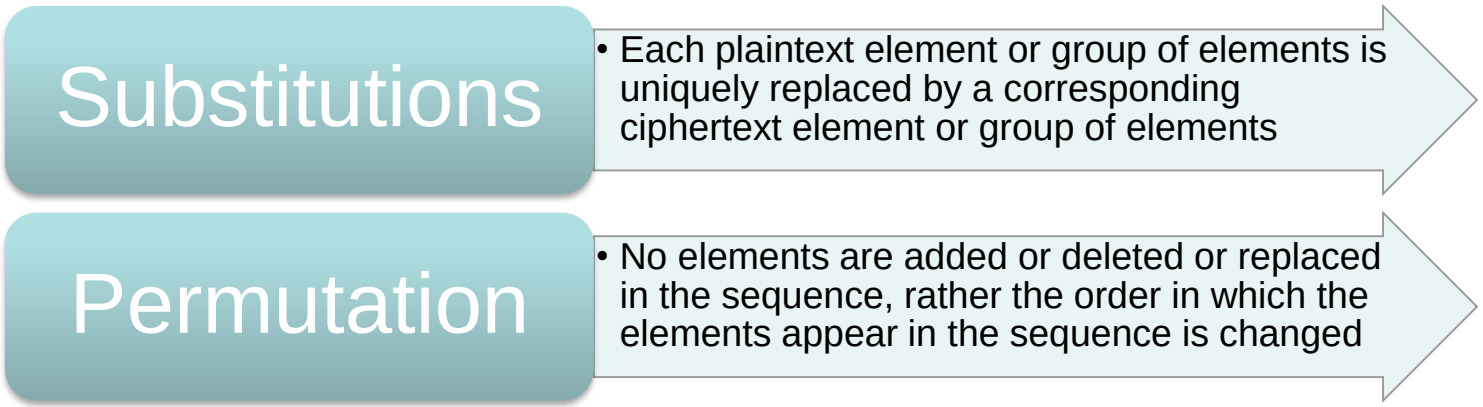
Ciphertext	Plaintext
0000	1110
0001	0011
0010	0100
0011	1000
0100	0001
0101	1100
0110	1010
0111	1111
1000	0111
1001	1101
1010	1001
1011	0110
1100	1011
1101	0010
1110	0000
1111	0101

Block Cipher Principles

- most symmetric block ciphers are based on a **Feistel Cipher Structure**
- needed since must be able to **decrypt** ciphertext to recover messages efficiently
- block ciphers look like an extremely large substitution
- would need table of 2^{64} entries for a 64-bit block
- instead create from smaller building blocks
- using idea of a product cipher

Feistel Cipher

- Feistel proposed the use of a cipher that alternates substitutions and permutations



  - Is a practical application of a proposal by Claude Shannon to develop a product cipher that alternates confusion and diffusion functions
  - Is the structure used by many significant symmetric block ciphers currently in use

Diffusion and Confusion

- Terms introduced by Claude Shannon to capture the two basic building blocks for any cryptographic system
  - Shannon's concern was to thwart cryptanalysis based on statistical analysis

Diffusion

- The statistical structure of the plaintext is dissipated into long-range statistics of the ciphertext
- This is achieved by having each plaintext digit affect the value of many ciphertext digits

Confusion

- Seeks to make the relationship between the statistics of the ciphertext and the value of the encryption key as complex as possible
- Even if the attacker can get some handle on the statistics of the ciphertext, the way in which the key was used to produce that ciphertext is so complex as to make it difficult to deduce the key

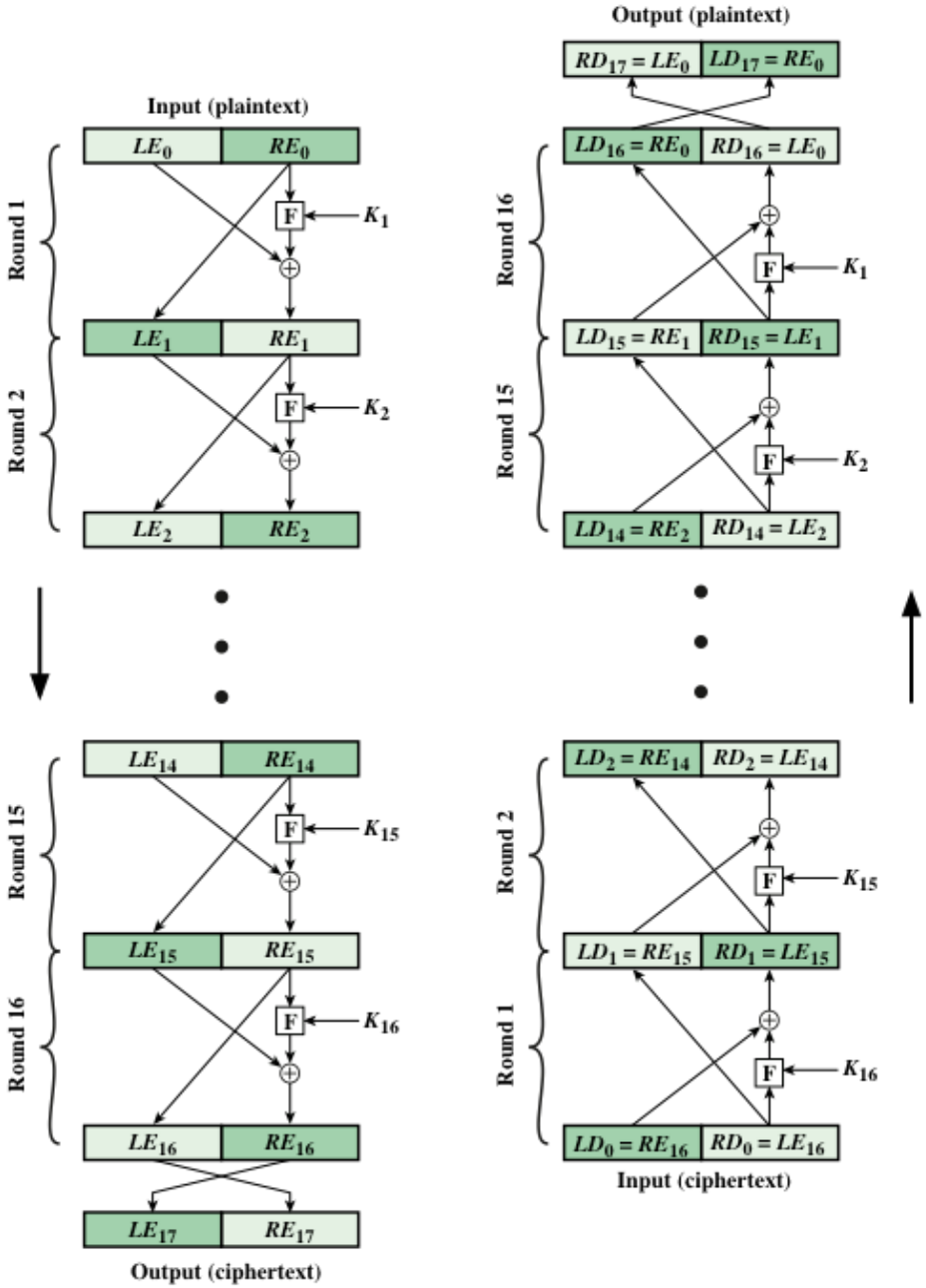


Figure 4.3 Feistel Encryption and Decryption (16 rounds)

Feistel Cipher Design Features

- Block size
  - Larger block sizes mean greater security but reduced encryption/decryption speed for a given algorithm
- Key size
  - Larger key size means greater security but may decrease encryption/decryption speeds
- Number of rounds
  - The essence of the Feistel cipher is that a single round offers inadequate security but that multiple rounds offer increasing security
- Subkey generation algorithm
  - Greater complexity in this algorithm should lead to greater difficulty of cryptanalysis
- Round function F
  - Greater complexity generally means greater resistance to cryptanalysis
- Fast software encryption/decryption
  - In many cases, encrypting is embedded in applications or utility functions in such a way as to preclude a hardware implementation; accordingly, the speed of execution of the algorithm becomes a concern
- Ease of analysis
  - If the algorithm can be concisely and clearly explained, it is easier to analyze that algorithm for cryptanalytic vulnerabilities and therefore develop a higher level of assurance as to its strength

Data Encryption Standard (DES)

- Issued in 1977 by the National Bureau of Standards (now NIST) as Federal Information Processing Standard 46
- Was the most widely used encryption scheme until the introduction of the Advanced Encryption Standard (AES) in 2001
- Algorithm itself is referred to as the Data Encryption Algorithm (DEA)
  - Data are encrypted in 64-bit blocks using a 56-bit key
  - The algorithm transforms 64-bit input in a series of steps into a 64-bit output
  - The same steps, with the same key, are used to reverse the encryption

DES History

- IBM developed Lucifer cipher
  - by team led by Feistel in late 60's
  - used 64-bit data blocks with 128-bit key
- then redeveloped as a commercial cipher with input from NSA and others
- in 1973 NBS issued request for proposals for a national cipher standard
- IBM submitted their revised Lucifer which was eventually accepted as the DES

DES Design Controversy

- although DES standard is public
- was considerable controversy over design
  - in choice of 56-bit key (vs Lucifer 128-bit)
  - and because design criteria were classified
- subsequent events and public analysis show in fact design was appropriate
- use of DES has flourished
  - especially in financial applications
  - still standardised for legacy application use

A Simple DES (S-DES)

- Key: 10 bits ($k=k_1k_2---k_9k_{10}$)
- Block size: 8 bits

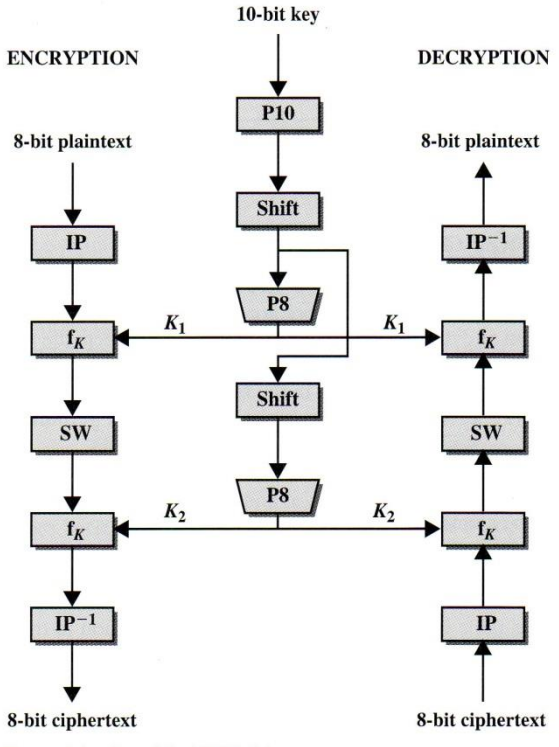


Figure: Simplified DES Scheme

A Simplified DES (S-DES)

- Key generation: input: 10bit key
  outputs: 8-bit k_1 and 8-bit k_2

K_1=P8(shift(P10(key)))

K_2=P8(Shift(shift(P10(key))))

where

P8(k_1k_2---k_9k_{10})=($k_6k_3k_7k_4k_8k_5k_{10}k_9$)

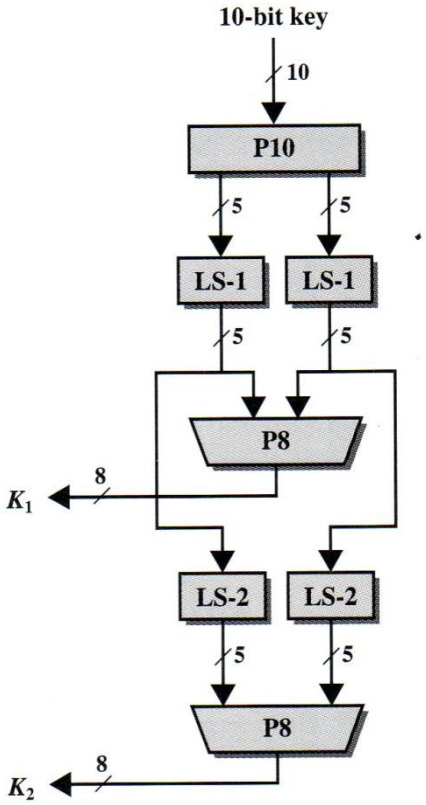
P10(k_1k_2---k_9k_{10})=($k_3k_5k_2k_7k_4k_{10}k_1k_9k_8k_6$)

shift(LS-1)is a circular left Shift

one bit operation

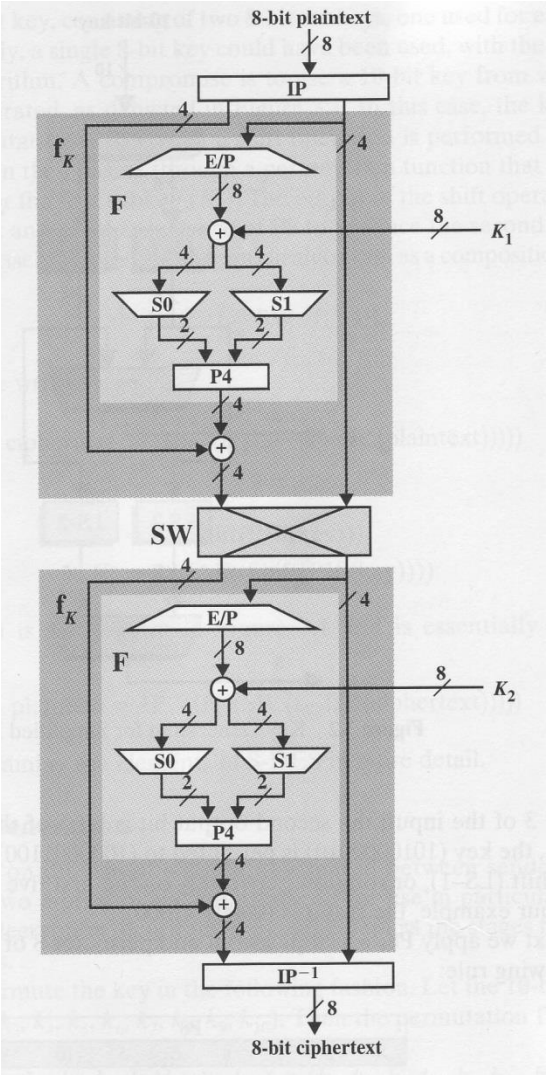
Shift(LS-2)is a circular left Shift

two bit operation



A Simplified DES (S-DES)

- encryption: key=(k_1,k_2)
  plaintext: 8-bit
  ciphertext:8-bit



Simplified DES Scheme Encryption Detail

A Simplified DES (S-DES)

- ciphertext=$IP^{-1}(f_{k2}(sw(f_{k1}(IP(plaintext)))))$

where

1. Initial permutation (IP):
   $IP(b_1b_2b_3b_4b_5b_6b_7b_8)=(b_2b_6b_3b_1b_4b_8b_5b_7)=(L,R)$
2. Inverse of IP (IP^{-1}):
   $IP^{-1}(b_1b_2b_3b_4b_5b_6b_7b_8)=(b_4b_1b_3b_5b_7b_2b_8b_6)$
   note: $IP^{-1}(IP(x))=x$
3. function f_k: inputs: 8-bit (L,R)
   outputs: 8-bit
   $f_k(L,R)=(L\oplus F(R,SK),R)$ where SK is a subkey
   function F: inputs: $R=(n_1n_2n_3n_4)$(4-bit)
   $SK=k_1=(k_{11}k_{12}k_{13}k_{14}k_{15}k_{16}k_{17}k_{18})$ (8-bit)

A Simplified DES (S-DES)

Step 1: generate 8 bits by R and SK:

P_{00}, P_{01}, P_{02}, P_{03}
P_{10}, P_{11}, P_{12}, P_{13}

$P_{00}=n_4 \oplus k_{11}$, $P_{01}=n_1 \oplus k_{12}$, $P_{02}=n_2 \oplus k_{13}$, $P_{03}=n_3 \oplus k_{14}$
$P_{10}=n_2 \oplus k_{15}$, $P_{11}=n_3 \oplus K_{16}$, $P_{12}=n_4 \oplus K_{17}$, $P_{13}=n_1 \oplus K_{18}$

Step 2:

- use (P_{00} P_{03}) and (P_{01} P_{02}) as row and column index to access a 2-bit number. (a_1 a_2) in S-box S0
- Use (P_{10} P_{13}) and (P_{11} P_{12}) as row and column index to access a 2-bit number (a_3 a_4) in S-box S1

$$S0=\begin{bmatrix}1 & 0 & 3 & 2\\ 3 & 2 & 1 & 0\\ 0 & 2 & 1 & 3\\ 3 & 1 & 3 & 2\end{bmatrix} \qquad S1=\begin{bmatrix}0 & 1 & 2 & 3\\ 2 & 0 & 1 & 3\\ 3 & 0 & 1 & 0\\ 2 & 1 & 0 & 3\end{bmatrix}$$

Step 3:

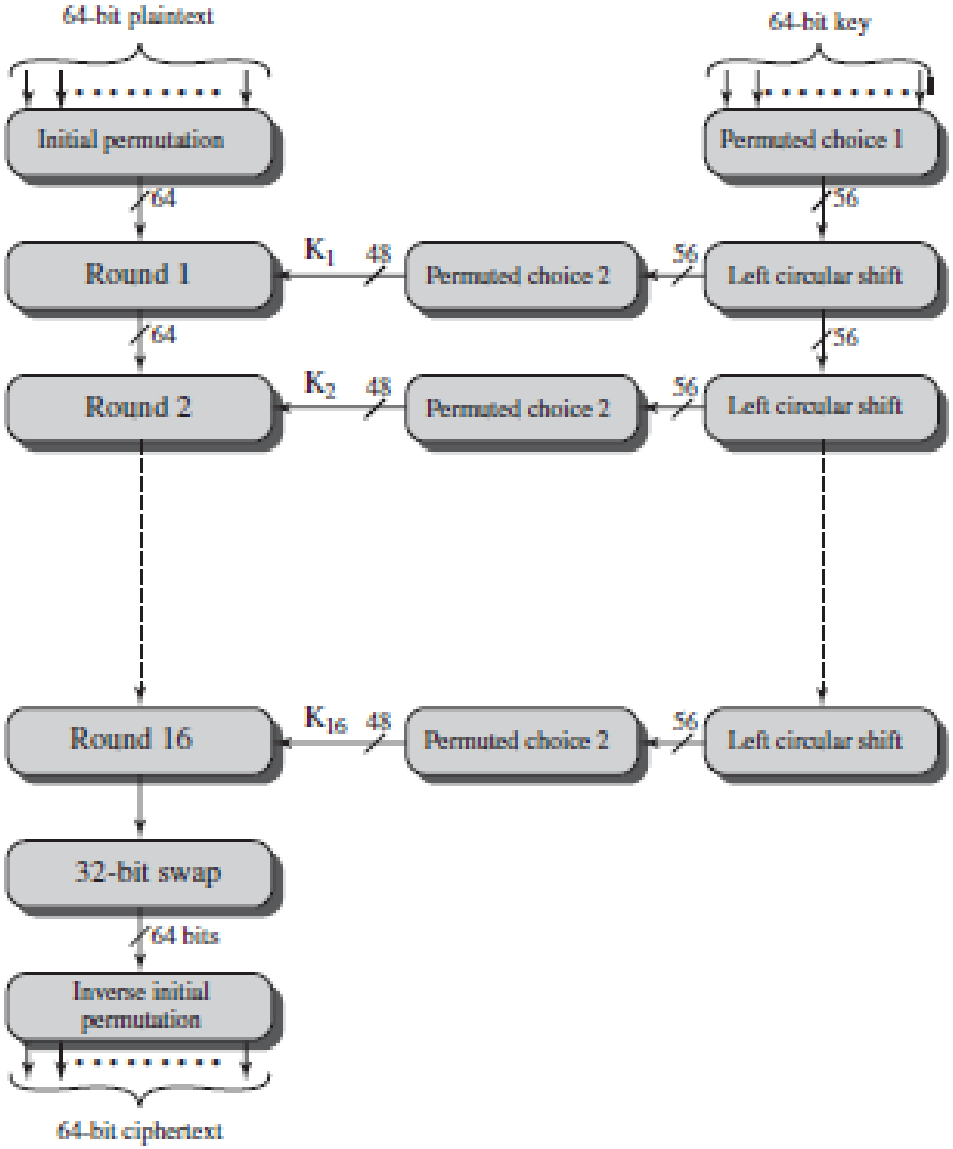
output of function F: P4=(a_2, a_4, a_3, a_1) (4-bit)

A Simplified DES (S-DES)

4. Switch function (SW)

   SW(L,R)=(R,L)

DES Encryption Overview



DES Key Schedule

- input: 64 bit key

  output: 16 48-bit subkeys: k_1, k_2, ---, k_{16}

  step1: pick up 56-bit key according to following table

(a) Input Key

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

  step2: perform permuted choice one (PC-1), the outputs are C_0 and D_0 (28-bit each)

(b) Permuted Choice One (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

DES Key Schedule

step3 for i from 1 to 16 do

C_i=shift(C_{i-1}) by the number of bits of table d

D_i=shift(D_{i-1}) by the number of bits of table d

(d) Schedule of Left Shifts

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

Step4 K_i = PC-2 (C_i), PC-2 (Permuted Choice Two) is defined by

(c) Permuted Choice Two (PC-2)

14	17	11	24	1	5	3	28
15	6	21	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	53	46	42	50	36	29	32

DES Encryption

Step1: Initial Permutation (IP)

IP(m_1, m_2, ---, m_{64})=(m_1', m_2', ---, m_{64}')

=(L_0, R_0) defined by

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

DES Encryption

Step2: 16 rounds (i=1 to 16)

In each round:

$L_i=R_{i-1}$

$R_i=L_{i-1} \oplus F(R_{i-1},K_i)$

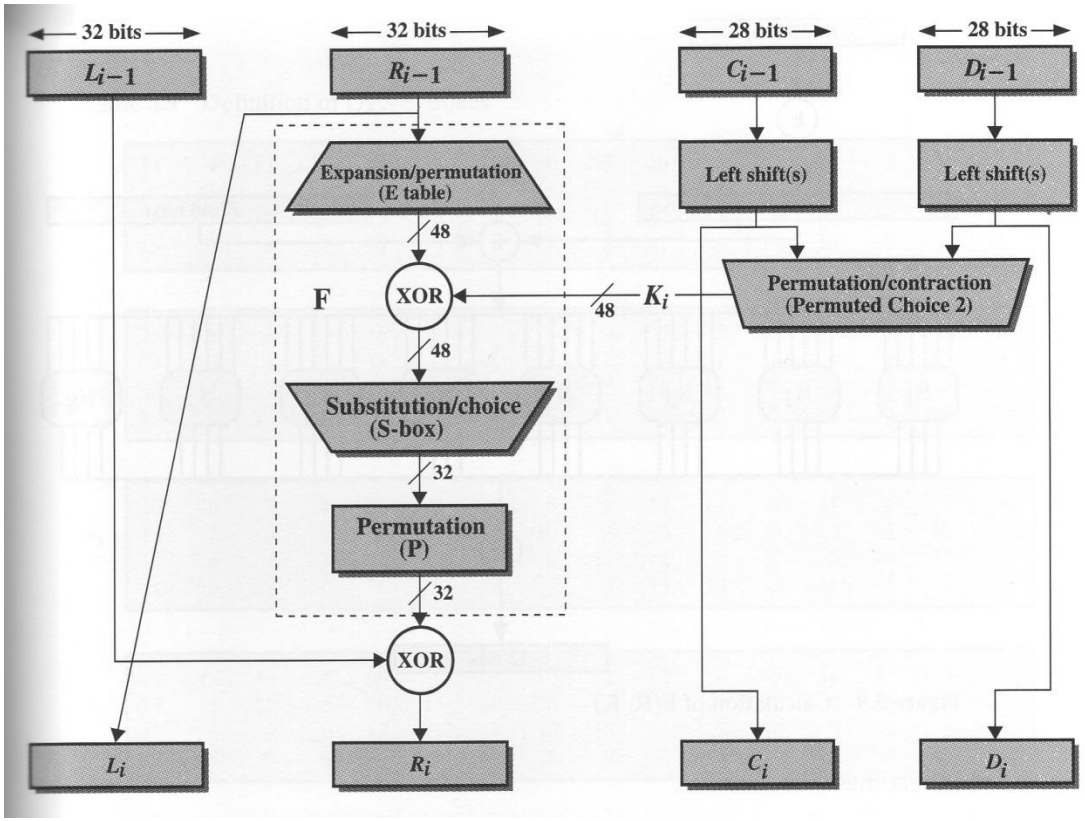


Figure: Single Round of DES Algorithm

DES Round Operations

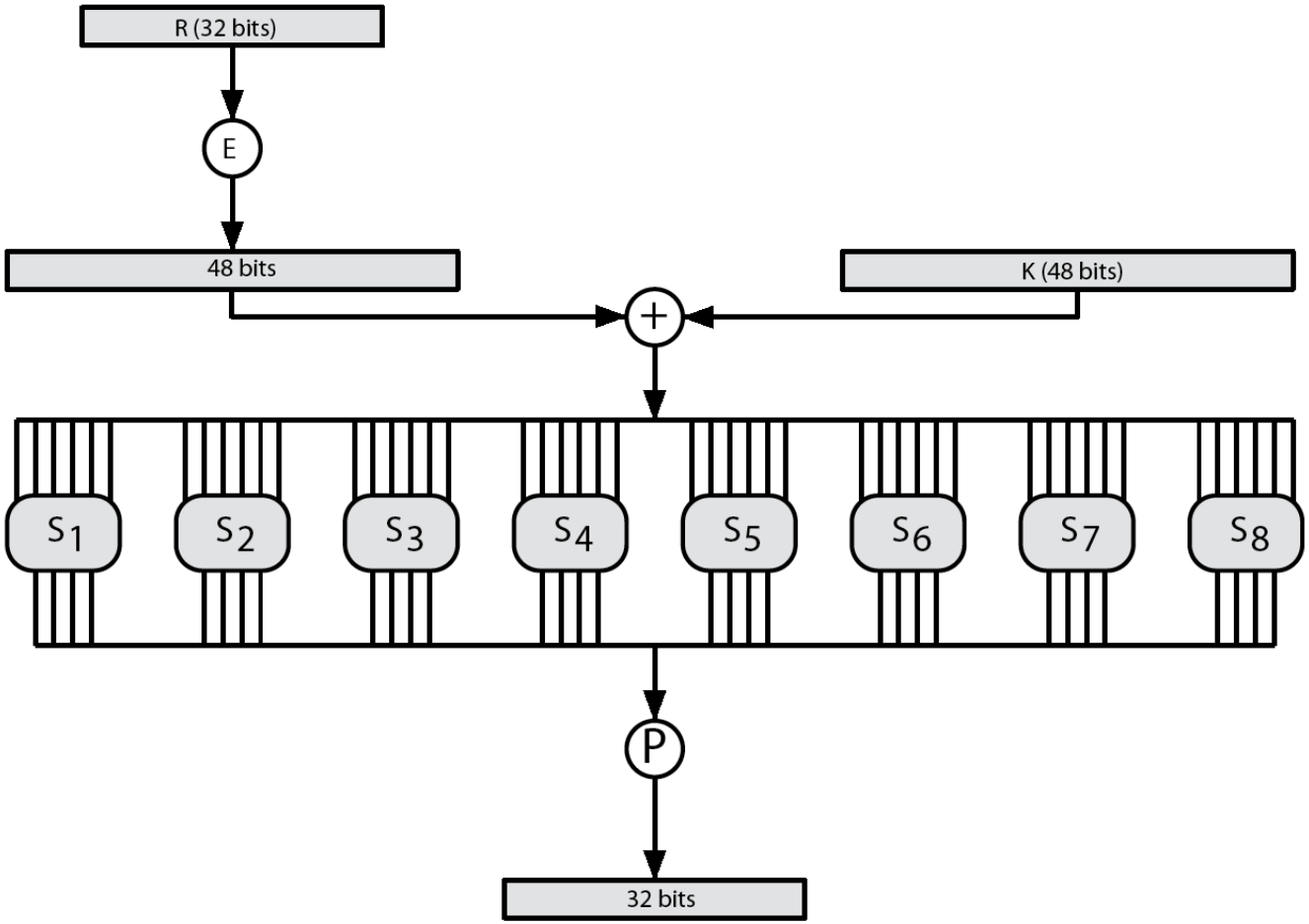
E (expansion/ permutation):

32-bit ->48-bits

(c) Expansion Permutation (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

DES Round Structure



Substitution Boxes S

- have eight S-boxes which map 6 to 4 bits
- each S-box is actually 4 little 4 bit boxes
  - outer bits 1 & 6 (**row** bits) select one row of 4
  - inner bits 2-5 (**col** bits) are substituted
  - result is 8 lots of 4 bits, or 32 bits
- row selection depends on both data & key
  - feature known as autoclaving (autokeying)
- example:
  - `S(18 09 12 3d 11 17 38 39) = 5fd25e03`

DES Encryption

Step3: 32-bit swap

Step4: Inverse Initial Permutation (IP^{-1})

(b) Inverse Initial Permutation (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

DES Decryption

- decrypt must unwind steps of data computation
- with Feistel design, do encryption steps again using subkeys in reverse order (SK16 … SK1)
  - IP undoes final FP step of encryption
  - 1st round with SK16 undoes 16th encrypt round
  - ….
  - 16th round with SK1 undoes 1st encrypt round
  - then final FP undoes initial encryption IP
  - thus recovering original data value

Avalanche Effect

- A desirable property of encryption algorithm
- where a change of **one** input or key bit results in changing approx. **half** output bits
- making attempts to “home-in” by guessing keys impossible
- DES shows strong avalanche effect

Avalanche Effect - Example

Change in Plaintext:

- P_1: **0**0000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
- P_2: **1**0000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
- Key: 0000001 1001011 0100100 1100010 0011100 0011000 0011100 0110010

Change in Key:

- P: 01101000 10000101 00101111 01111010 00010011 01110110 11101011 10100100
- Key_1: **1**110010 1111011 1101111 0011000 0011101 0000100 0110001 11011100
- Key_2: **0**110010 1111011 1101111 0011000 0011101 0000100 0110001 11011100

Avalanche Effect - Example

(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

Strength of DES – Key Size

- 56-bit keys have $2^{56} = 7.2 \times 10^{16}$ values
- brute force search looks hard
  - but actually not really
  - 1997 on a large network of computers in a few months
  - in 1998 on dedicated h/w (EFF) in a few days
  - in 1999 above combined in 22hrs!
- still must be able to recognize plaintext
- alternatives to DES

Strength of DES – Analytic Attacks

- have several analytic attacks on DES
- these utilise some deep structure of the cipher
  - by gathering information about encryptions
  - can eventually recover some/all of the sub-key bits
  - if necessary then exhaustively search for the rest
- generally these are statistical attacks
- include
  - differential cryptanalysis
  - linear cryptanalysis
  - related key attacks

Timing Attacks

- attacks actual implementation of cipher
- one in which information about the key or the plaintext is obtained by observing how long it takes a given implementation to perform decryptions on various ciphertexts
- exploits the fact that an encryption or decryption algorithm often takes slightly different amounts of time on different inputs
- so far it appears unlikely that this technique will ever be successful against DES or more powerful symmetric ciphers such as triple DES and AES
- particularly problematic on smartcards

Differential Cryptanalysis

- one of the most significant recent (public) advances in cryptanalysis
- known by NSA in 70's cf DES design
- Murphy, Biham & Shamir published in 90’s
- powerful method to analyse block ciphers
- used to analyse most current block ciphers with varying degrees of success
- DES reasonably resistant to it, cf Lucifer

Differential Cryptanalysis

- a statistical attack against Feistel ciphers
- uses cipher structure not previously used
- design of S-P networks has output of function *f* influenced by both input & key
- hence cannot trace values back through cipher without knowing value of the key
- differential cryptanalysis compares two related pairs of encryptions

Differential Cryptanalysis

- This analysis compares **differences** between two related encryptions, and looks for a **known difference in** leading to a **known difference out** with some (pretty small but still significant) probability. If a number of such differences are determined, it is feasible to determine the subkey used in the function f.

Linear Cryptanalysis

- another recent development
- also a statistical method
- must be iterated over rounds, with decreasing probabilities
- developed by Matsui et al in early 90's
- based on finding linear approximations
- can attack DES with 2^{43} known plaintexts, easier but still in practise infeasible

DES Design Criteria

- as reported by Coppersmith in [COPP94]
- 7 criteria for S-boxes provide for
  - non-linearity
  - resistance to differential cryptanalysis
  - good confusion
- 3 criteria for permutation P provide for
  - increased diffusion

Block Cipher Design

- basic principles still like Feistel's in 1970's
- number of rounds
- function f:
- key schedule

Block Cipher Design Principles: Number of Rounds

The greater the number of rounds, the more difficult it is to perform cryptanalysis

In general, the criterion should be that the number of rounds is chosen so that known cryptanalytic efforts require greater effort than a simple brute-force key search attack

If DES had 15 or fewer rounds, differential cryptanalysis would require less effort than a brute-force key search

Block Cipher Design Principles: Design of Function F

- The heart of a Feistel block cipher is the function F
- The more nonlinear F, the more difficult any type of cryptanalysis will be
- The SAC and BIC criteria appear to strengthen the effectiveness of the confusion function

The algorithm should have good avalanche properties

Strict avalanche criterion (SAC)

States that any output bit j of an S-box should change with probability 1/2 when any single input bit i is inverted for all i , j

Bit independence criterion (BIC)

States that output bits j and k should change independently when any single input bit i is inverted for all i , j , and k

Block Cipher Design Principles: Key Schedule Algorithm

- With any Feistel block cipher, the key is used to generate one subkey for each round
- In general, we would like to select subkeys to maximize the difficulty of deducing individual subkeys and the difficulty of working back to the main key
- It is suggested that, at a minimum, the key schedule should guarantee key/ciphertext Strict Avalanche Criterion and Bit Independence Criterion

Multiple Encryption & DES

- clear a replacement for DES was needed
  - theoretical attacks that can break it
  - demonstrated exhaustive key search attacks
- AES is a new cipher alternative
- prior to this alternative was to use multiple encryption with DES implementations
- Triple-DES is the chosen form

Double-DES?

- could use 2 DES encrypts on each block
  - $C = E_{K2}(E_{K1}(P))$
- issue of reduction to single stage
- and have “meet-in-the-middle” attack
  - it is a known plaintext attack (e.g., know a pair (P, C))
  - works whenever use a cipher twice
  - since $X = E_{K1}(P) = D_{K2}(C)$
  - attack by encrypting P for all possible values of K_1, and store these results in a table
  - then decrypt C using all possible values of K_2 and check the result against the table for a match
  - if match occurs, and works for another pair, then key
  - can takes $O(2^{56})$ steps

Triple-DES with Two-Keys

- hence must use 3 encryptions
  - would seem to need 3 distinct keys
- but can use 2 keys with E-D-E sequence
  - $C = E_{K1}(D_{K2}(E_{K1}(P)))$
  - note: encrypt & decrypt equivalent in security
  - if $K_1=K_2$ then can work with single DES
- standardized in ANSI X9.17 & ISO8732
- no current known practical attacks

Triple-DES with Three-Keys

- although no practical attacks on two-key Triple-DES, user may feel some concern
- can use Triple-DES with Three-Keys
  - $C = E_{K3}(D_{K2}(E_{K1}(P)))$
- has been adopted by some Internet applications, eg PGP, S/MIME

Modes of Operation

- block ciphers encrypt fixed size blocks
  - eg. DES encrypts 64-bit blocks with 56-bit key
- need some way to en/decrypt arbitrary amounts of data in practise
- **ANSI X3.106-1983 Modes of Use** (now FIPS 81) defines 4 possible modes
- subsequently 5 block cipher modes of operation
- have **block** and **stream** modes

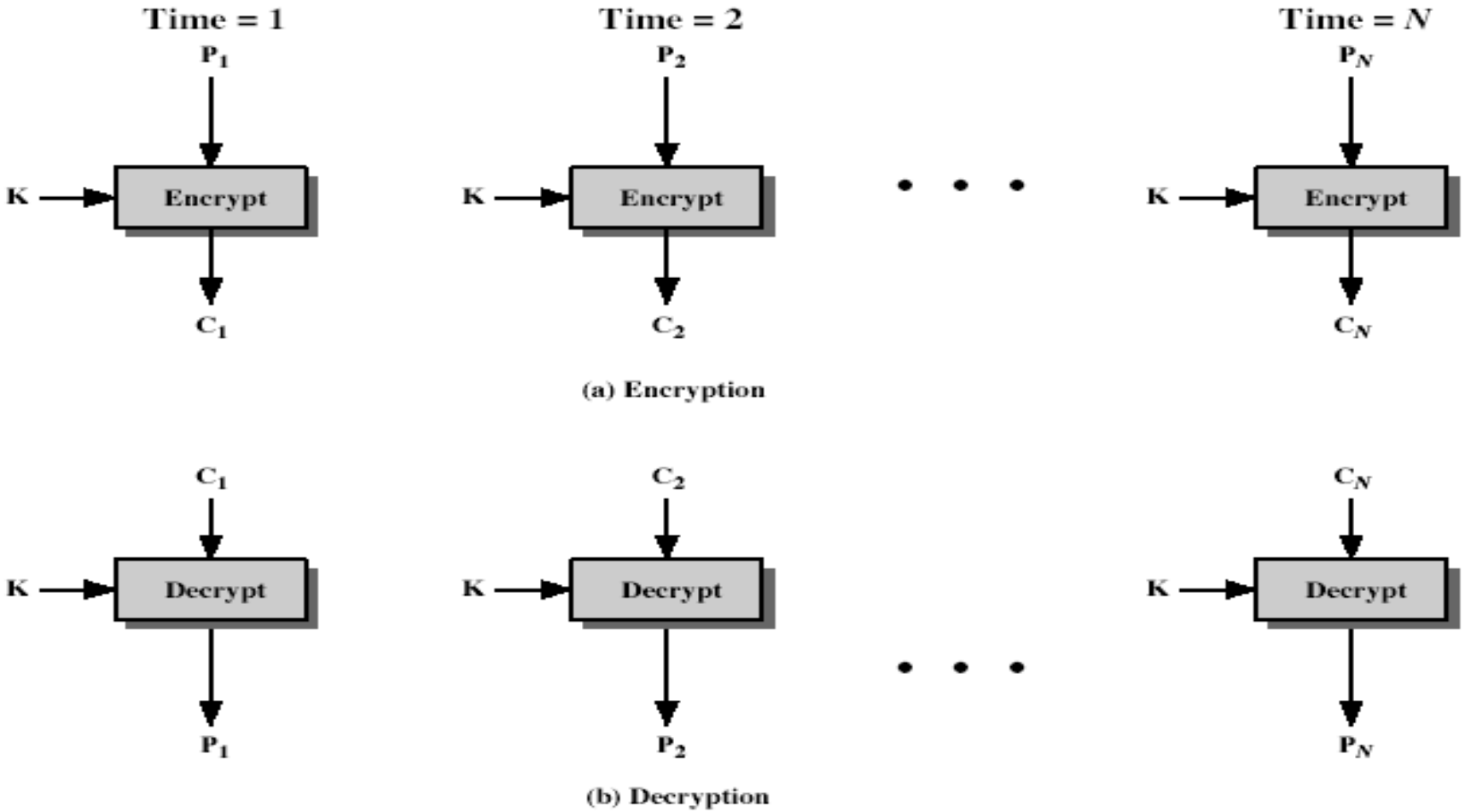
Electronic Codebook Book (ECB)

- message is broken into independent blocks which are encrypted
- each block is a value which is substituted, like a codebook, hence name
- each block is encoded independently of the other blocks

$$C_i = E_{K1}(P_i)$$

- uses: secure transmission of single values

Electronic Codebook Book (ECB)



Advantages and Limitations of ECB

- message repetitions may show in ciphertext
  - if aligned with message block
  - particularly with data such as graphics
  - or with messages that change very little, which become a code-book analysis problem
- weakness is due to the encrypted message blocks being independent
- main use is sending a few blocks of data

Cipher Block Chaining (CBC)

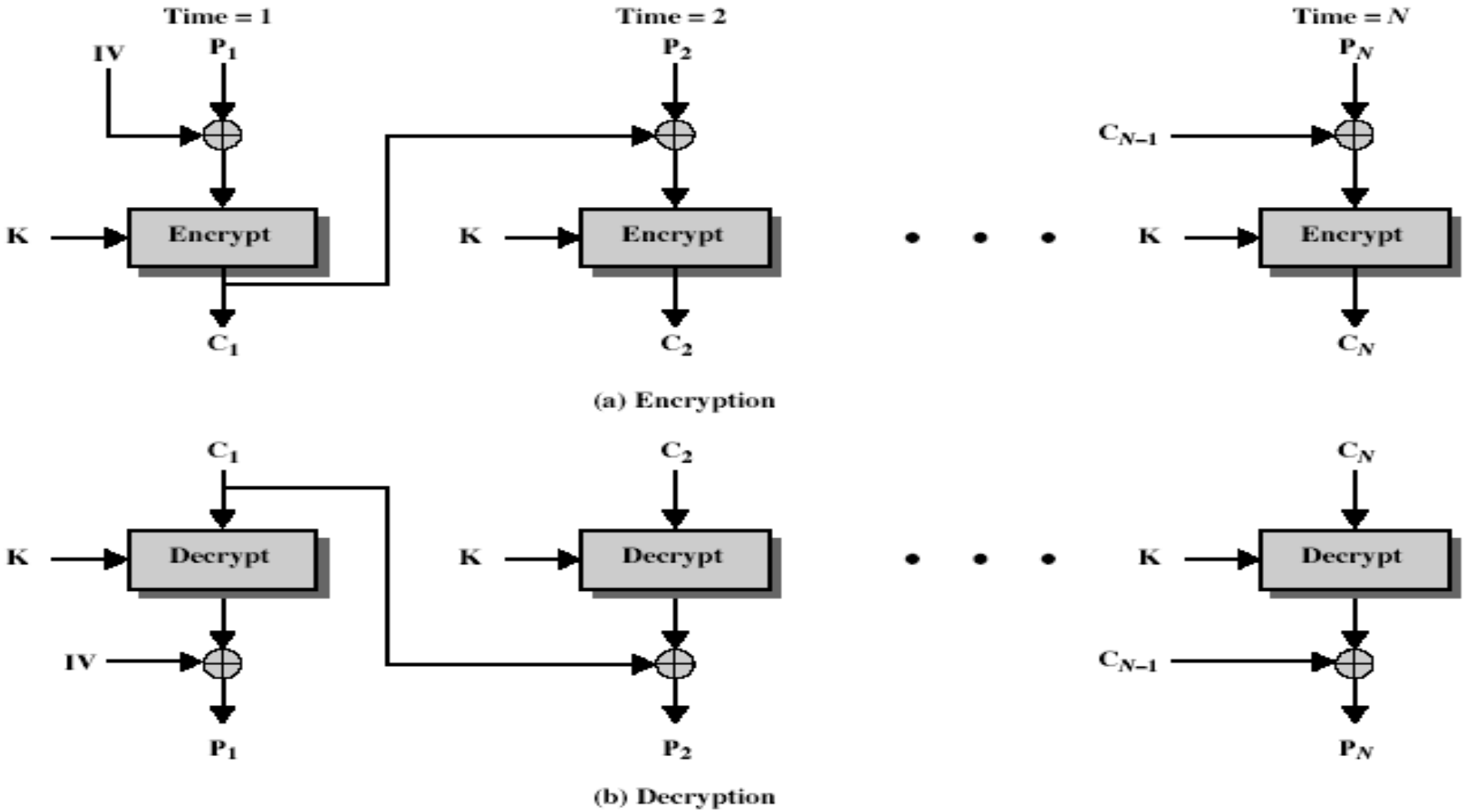
- message is broken into blocks
- linked together in encryption operation
- each previous cipher blocks is chained with current plaintext block, hence name
- use Initial Vector (IV) to start process

$$C_i = E_{K1}(P_i \text{ XOR } C_{i-1})$$

$$C_{-1} = IV$$

- uses: bulk data encryption, authentication

Cipher Block Chaining (CBC)



Message Padding

- at end of message must handle a possible last short block
  - which is not as large as blocksize of cipher
  - pad either with known non-data value (eg nulls)
  - or pad last block along with count of pad size
    - eg. [b1 b2 b3 0 0 0 0 5]
    - means have 3 data bytes, then 5 bytes pad+count
  - this may require an extra entire block over those in message
- there are other, more esoteric modes, which avoid the need for an extra block

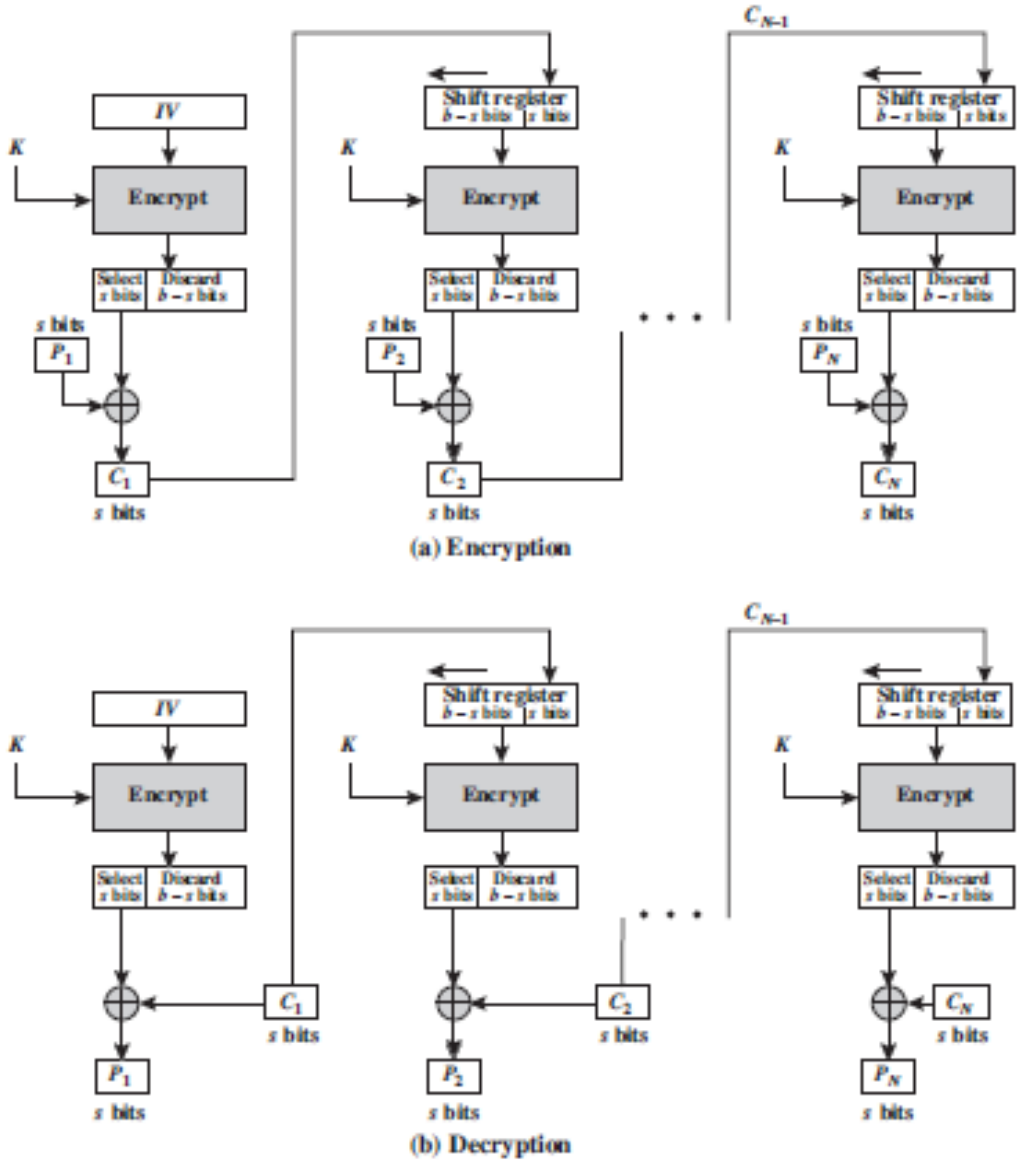
Advantages and Limitations of CBC

- a ciphertext block depends on **all** blocks before it
- any change to a block affects all following ciphertext blocks
- need **Initialization Vector** (IV)
  - which must be known to sender & receiver
  - if sent in clear, attacker can change bits of first block, and change IV to compensate
  - hence IV must either be a fixed value
  - or must be sent encrypted in ECB mode before the rest of the message

Cipher FeedBack (CFB)

- message is treated as a stream of bits
- added to the output of the block cipher
- result is feed back for next stage (hence name)
- standard allows any number of bit (1,8, 64 or 128 etc) to be feed back
  - denoted CFB-1, CFB-8, CFB-64, CFB-128 etc
- most efficient to use all bits in block (64 or 128)
- uses: stream data encryption, authentication

Cipher FeedBack (CFB)



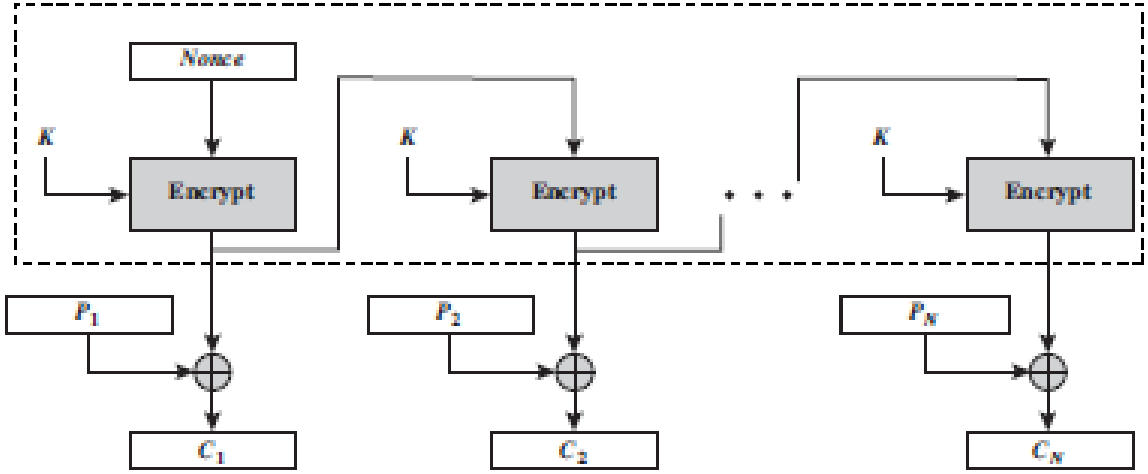
Advantages and Limitations of CFB

- appropriate when data arrives in bits/bytes
- most common stream mode
- limitation is need to stall while do block encryption after every n-bits
- note that the block cipher is used in **encryption** mode at **both** ends
- an error can cause issues

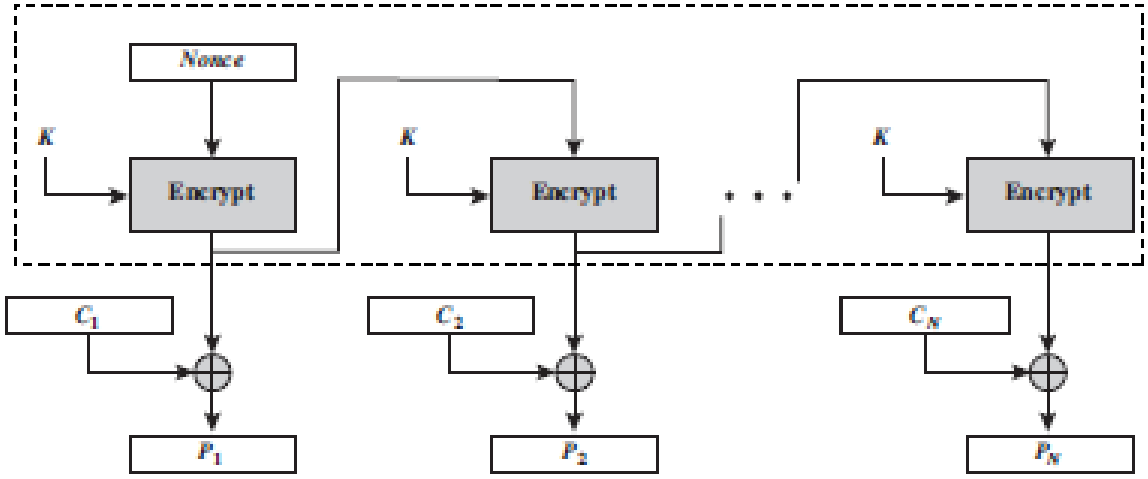
Output FeedBack (OFB)

- message is treated as a stream of bits
- output of cipher is added to message
- output is then feed back (hence name)
- feedback is independent of message
- uses: stream encryption on noisy channels

Output FeedBack (OFB)



(a) Encryption



(b) Decryption

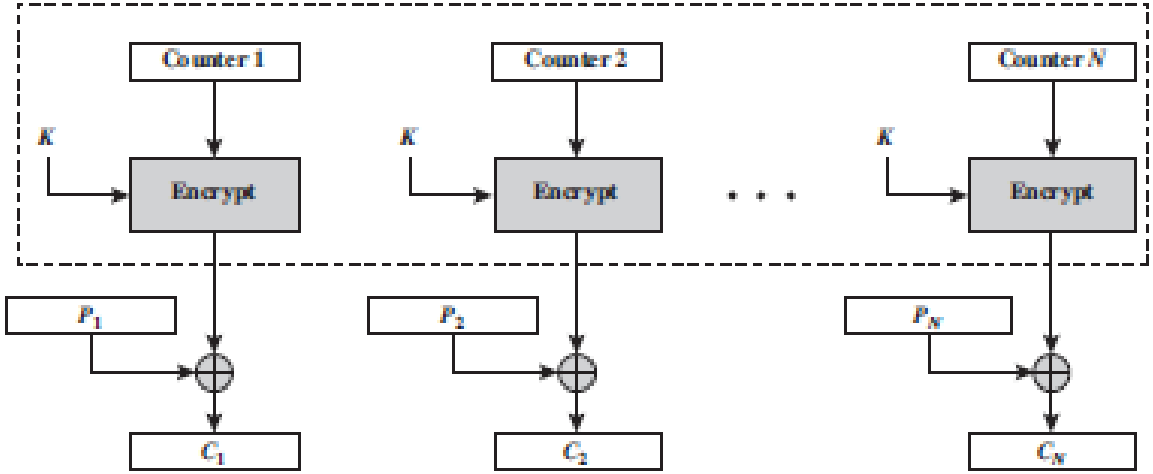
Advantages and Limitations of OFB

- an error can still cause issues?
- more vulnerable to message stream modification
- a variation of a Vernam cipher
  - hence must **never** reuse the same sequence (key+IV)
- sender & receiver must remain in sync
- originally specified with m-bit feedback
- subsequent research has shown that only **full block feedback** (ie CFB-64 or CFB-128) should ever be used

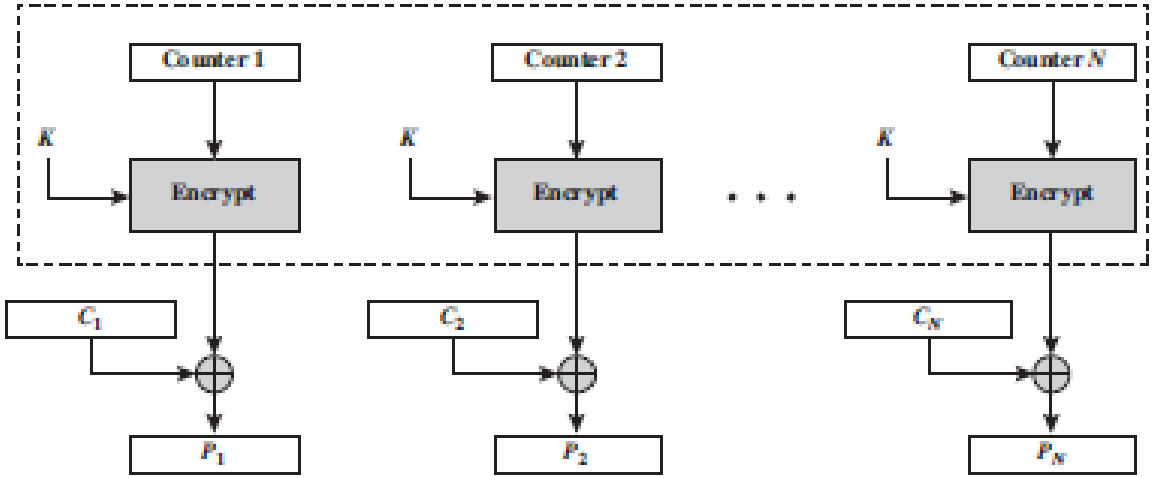
Counter (CTR)

- a “new” mode, though proposed early on
- similar to OFB but encrypts counter value rather than any feedback value
- must have a different key & counter value for every plaintext block (never reused)
- uses: high-speed network encryptions

Counter (CTR)



(a) Encryption



(b) Decryption

Advantages and Limitations of CTR

- efficiency
  - can do parallel encryptions in h/w or s/w
  - can preprocess in advance of need
  - good for bursty high speed links
- random access to encrypted data blocks
- provable security (good as other modes)
- but must ensure never reuse key/counter values, otherwise could break (cf OFB)