राष्ट्रीय न्यायिक विज्ञान विश्वविद्यालय
National Forensic Sciences University

गृह मंत्रालय
MINISTRY OF
HOME AFFAIRS

Network Security

Dr. Lokesh Chouhan

Dean Academics and Associate Professor

गृह मंत्रालय
MINISTRY OF
HOME AFFAIRS

राष्ट्रीय न्यायालयिक विज्ञान विश्वविद्यालय
(राष्ट्रीय महत्त्व का संस्थान, गृह मंत्रालय, भारत सरकार)

National Forensic Sciences University
(An Institution of National Importance under Ministry of Home Affairs, Government of India)

E-Mail: Lokeshchouhan@gmail.com, Lokesh.chouhan_goa@nfsu.ac.in

Mob: 9827235155

Polybius Square Cipher

- A Polybius Square is a table that allows someone to convert letters into numbers.
- To make the encryption little harder, this table can be randomized and shared with the recipient.
- In order to fit the 26 letters of the alphabet into the 25 cells created by the table, the letters 'i' and 'j' are usually combined into a single cell. Originally there was no such problem because the Greek alphabet has 24 letters.

Polybius Square Cipher

	1	2	3	4	5
1	A	B	C	D	E
2	F	G	H	I,J	K
3	L	M	N	O	P
4	Q	R	S	T	U
5	V	W	X	Y	Z

Polybius Square Cipher

	1	2	3	4	5
1	A	B	C	D	E
2	F	G	H	I,J	K
3	L	M	N	O	P
4	Q	R	S	T	U
5	V	W	X	Y	Z

- Plaintext:
- ICC

- Ciphertext
- 241313

- Plaintext
- World Cup
- ?

Cryptographic Techniques

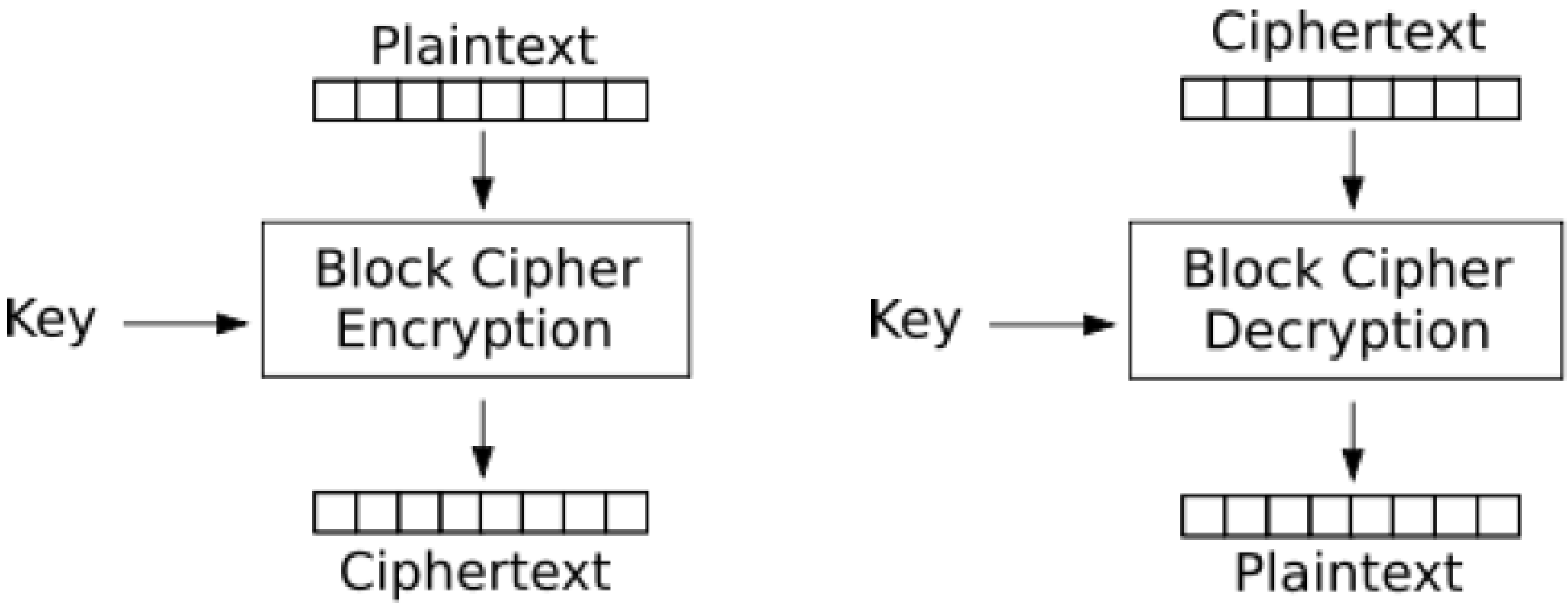
Modern Block Ciphers

- will now look at **modern block ciphers**
- one of the most widely used types of cryptographic algorithms
- provide secrecy and/or authentication services
- in particular will introduce DES (Data Encryption Standard)

Block vs Stream Ciphers

- block ciphers process messages in into 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
- hence are focus of course

Block Cipher



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**

Claude Shannon and Substitution-Permutation Ciphers

- in 1949 **Claude Shannon** introduced idea of **substitution-permutation (S-P) networks**
  - modern substitution-transposition product cipher
- these form the basis of modern block ciphers
- S-P networks are based on the two primitive cryptographic operations we have seen before:
  - *substitution* (S-box)
  - *permutation* (P-box)
- provide *confusion* and *diffusion* of message

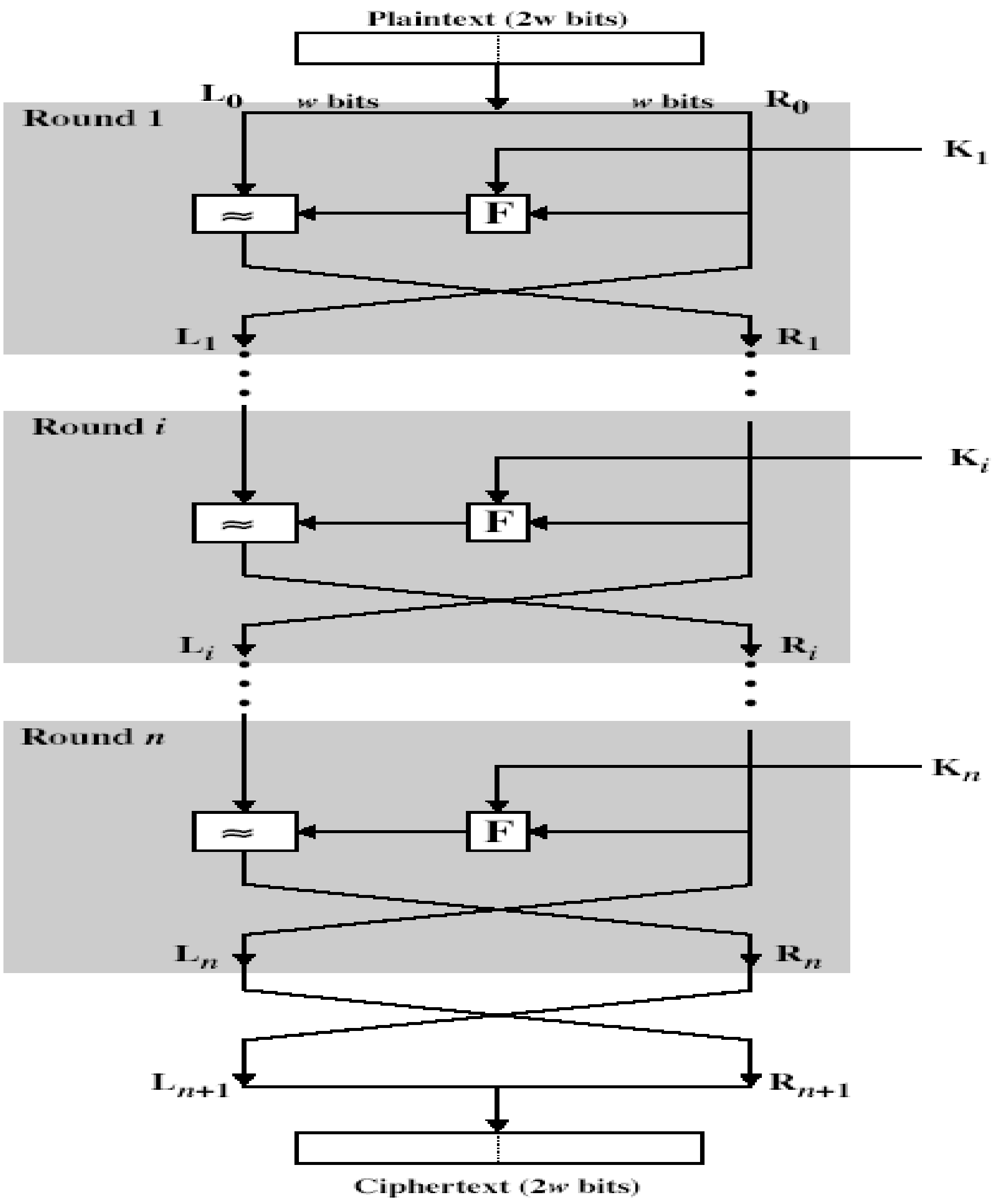
Confusion and Diffusion

- cipher needs to completely obscure statistical properties of original message
- a one-time pad does this
- more practically Shannon suggested combining elements to obtain:
- **diffusion** – dissipates statistical structure of plaintext over bulk of ciphertext
- **confusion** – makes relationship between ciphertext and key as complex as possible

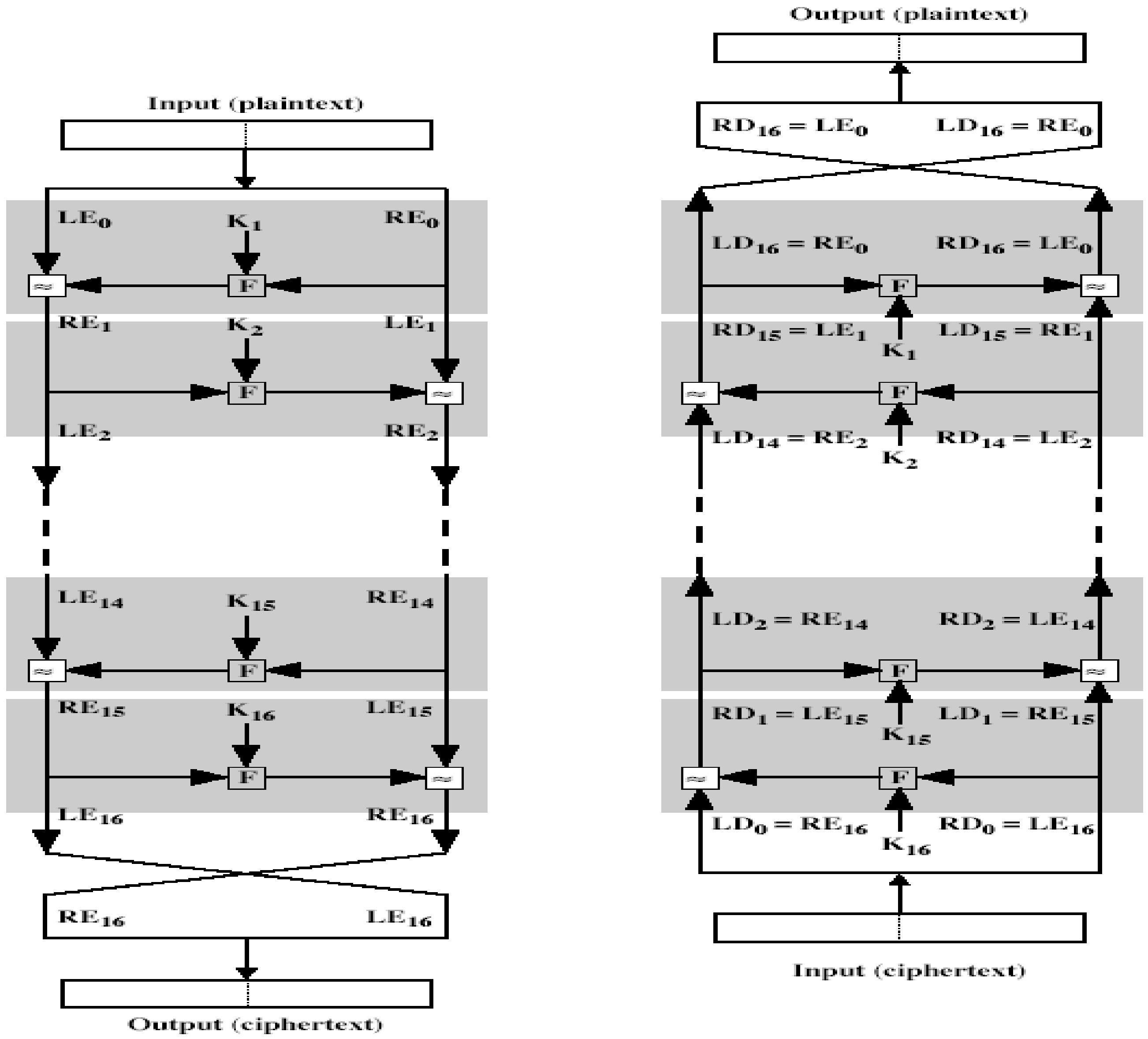
Feistel Cipher Structure

- Horst Feistel devised the **feistel cipher**
  - based on concept of invertible product cipher
- partitions input block into **two halves**
  - process through multiple rounds which
  - perform a substitution on left data half
  - based on round function of right half & subkey
  - then have permutation swapping halves
- implements Shannon's substitution-permutation (SP) network concept

Feistel Cipher Structure



Feistel Cipher Decryption



Feistel Cipher Design Principles

- **block size**
  - increasing size improves security, but slows cipher
- **key size**
  - increasing size improves security, makes exhaustive key searching harder, but may slow cipher
- **number of rounds**
  - increasing number improves security, but slows cipher
- **subkey generation**
  - greater complexity can make analysis harder, but slows cipher
- **round function**
  - greater complexity can make analysis harder, but slows cipher
- **fast software en/decryption & ease of analysis**
  - are more recent concerns for practical use and testing

Data Encryption Standard (DES)

- most widely used block cipher in world
- adopted in 1977 by NBS (now NIST)
  - as FIPS PUB 46
- encrypts 64-bit data using 56-bit key
- has widespread use
- has been considerable controversy over its security

DES History

- IBM developed Lucifer cipher
  - by team led by Feistel
  - 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

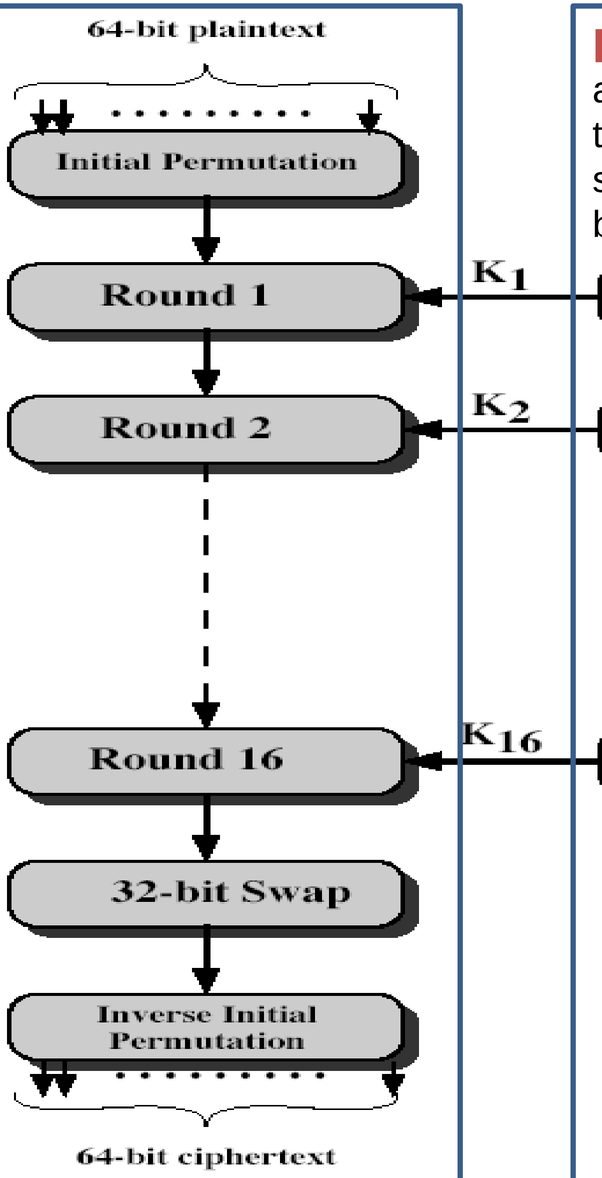
DES Design Controversy

- 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
- DES has become widely used, esp in financial applications

DES Encryption

Data

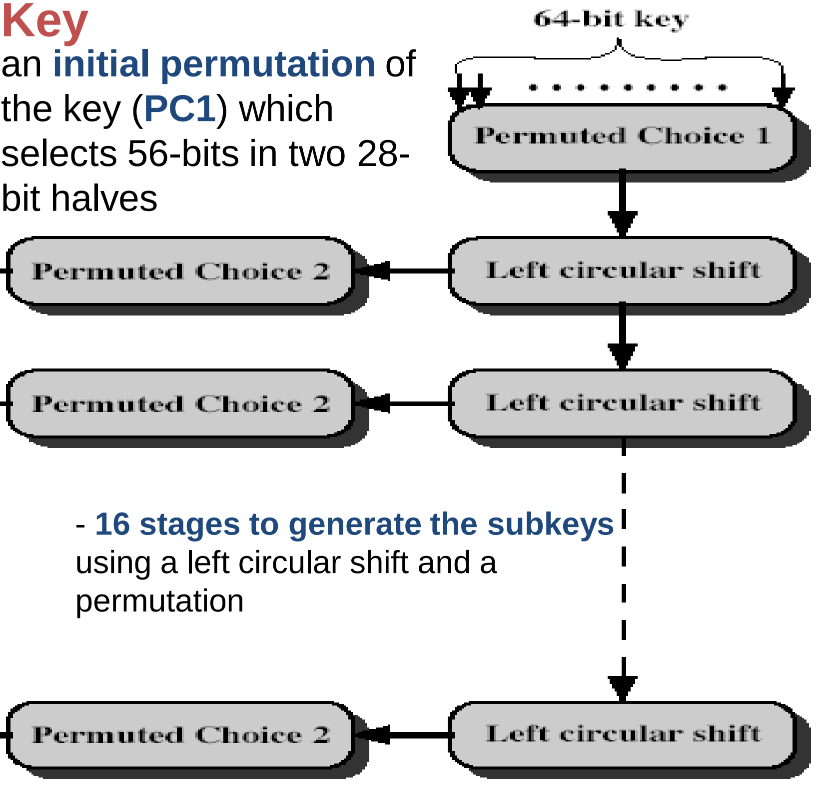
- an **initial permutation (IP)**
- **16 rounds** of a complex key dependent round function involving substitution and permutation functions
- a final permutation, being the **inverse of IP**



Key

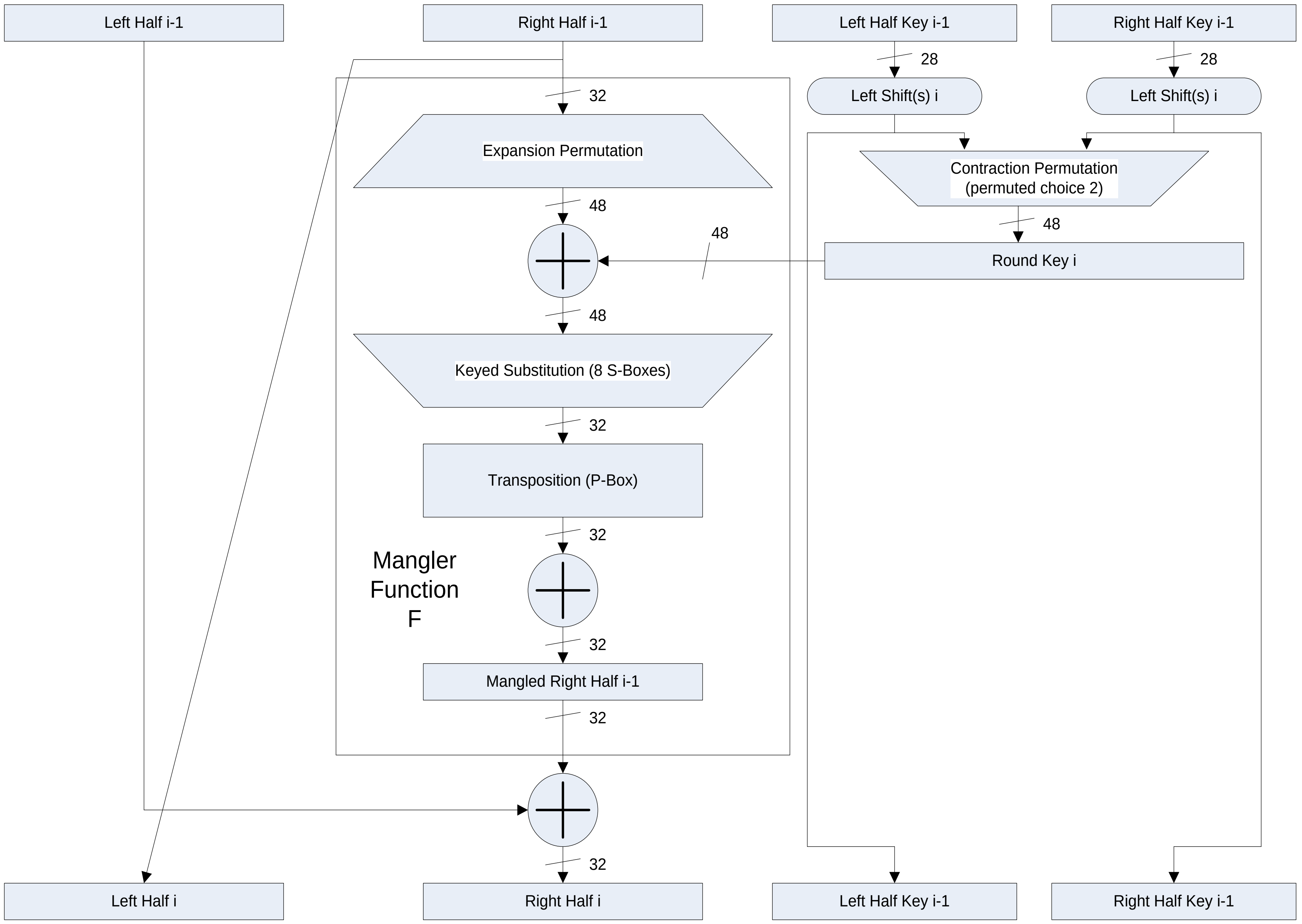
an **initial permutation** of the key (**PC1**) which selects 56-bits in two 28-bit halves

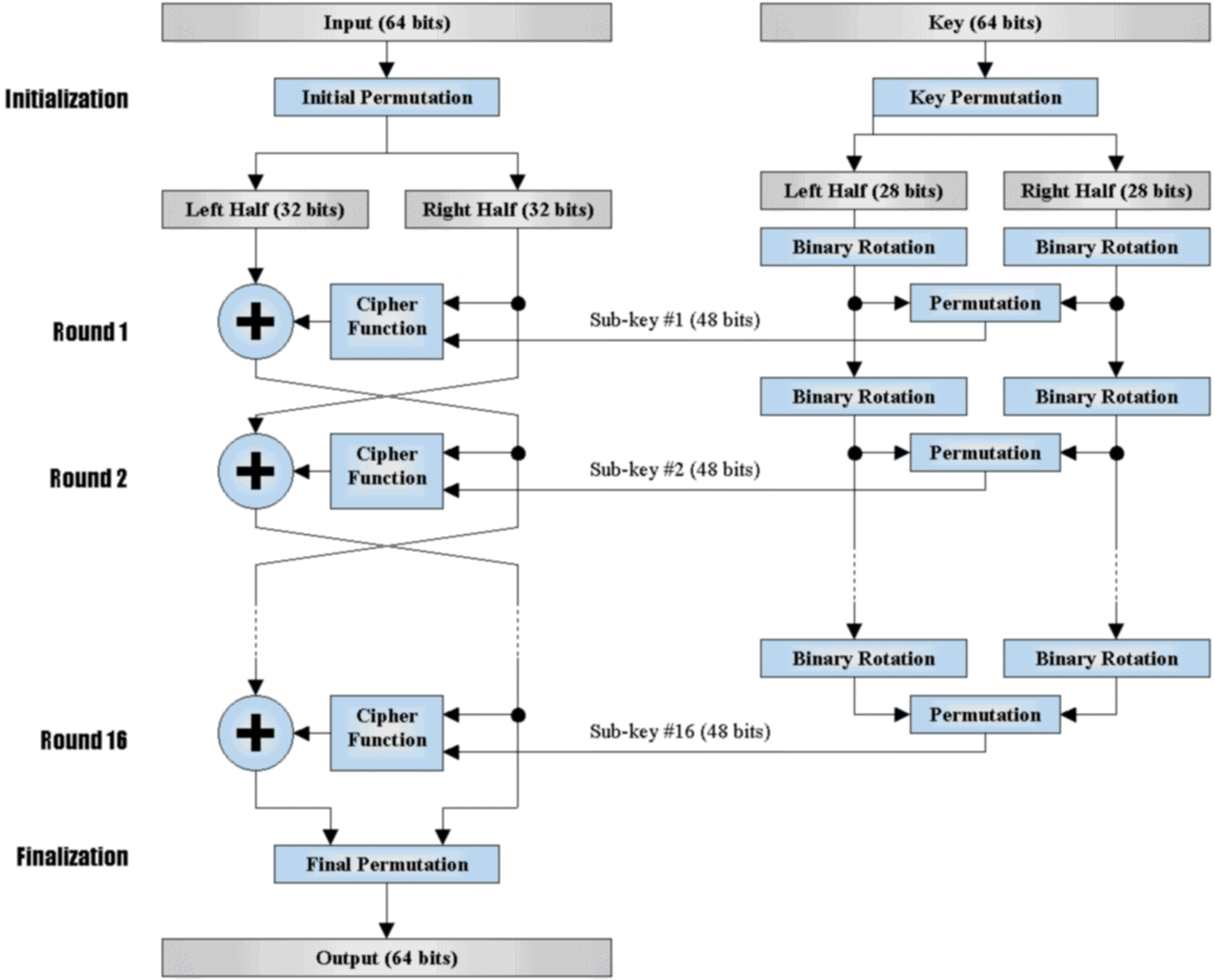
- **16 stages to generate the subkeys** using a left circular shift and a permutation

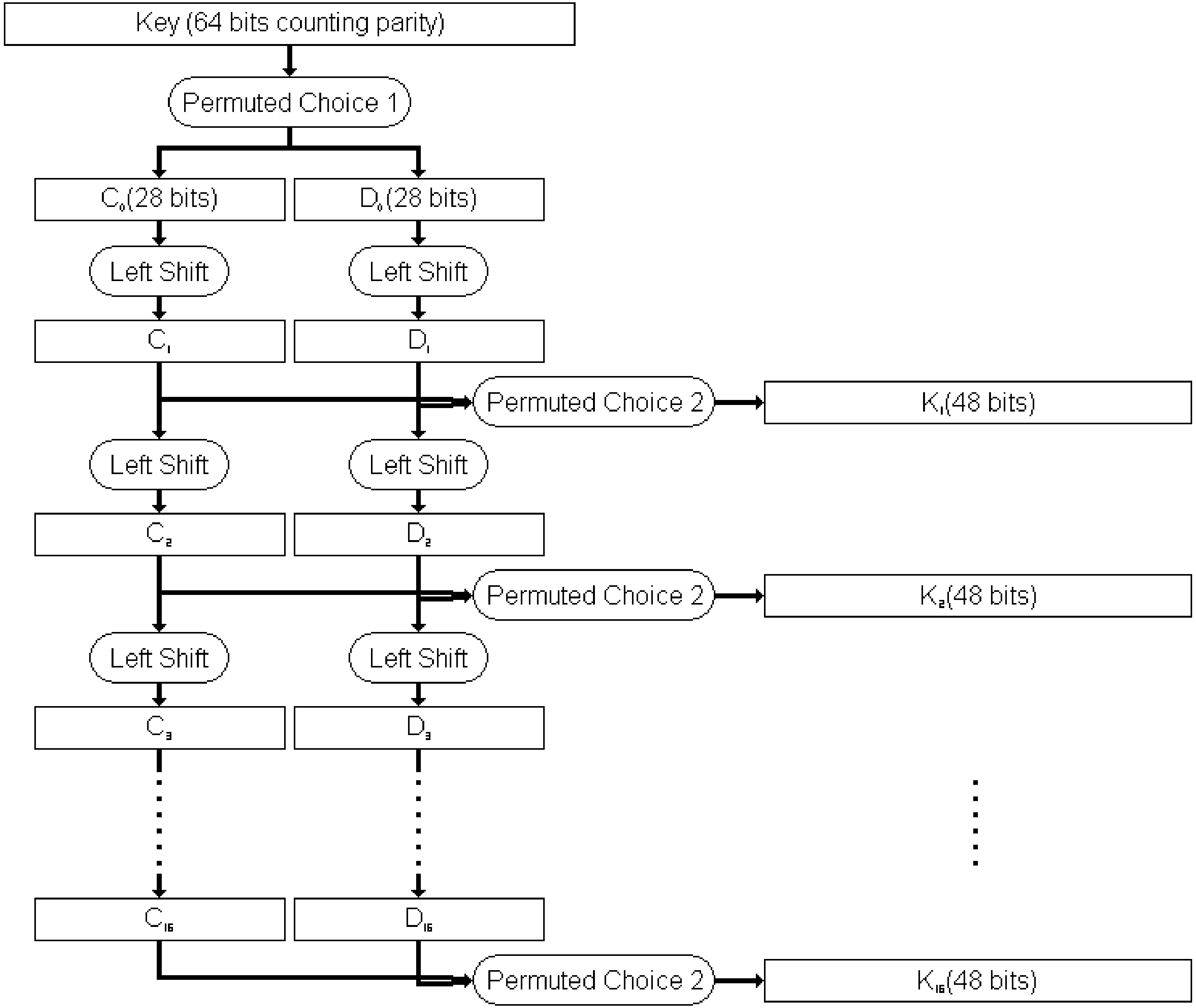


गृह मंत्रालय
MINISTRY OF
HOME AFFAIRS
सत्यमेव जयते

DES Round Structure



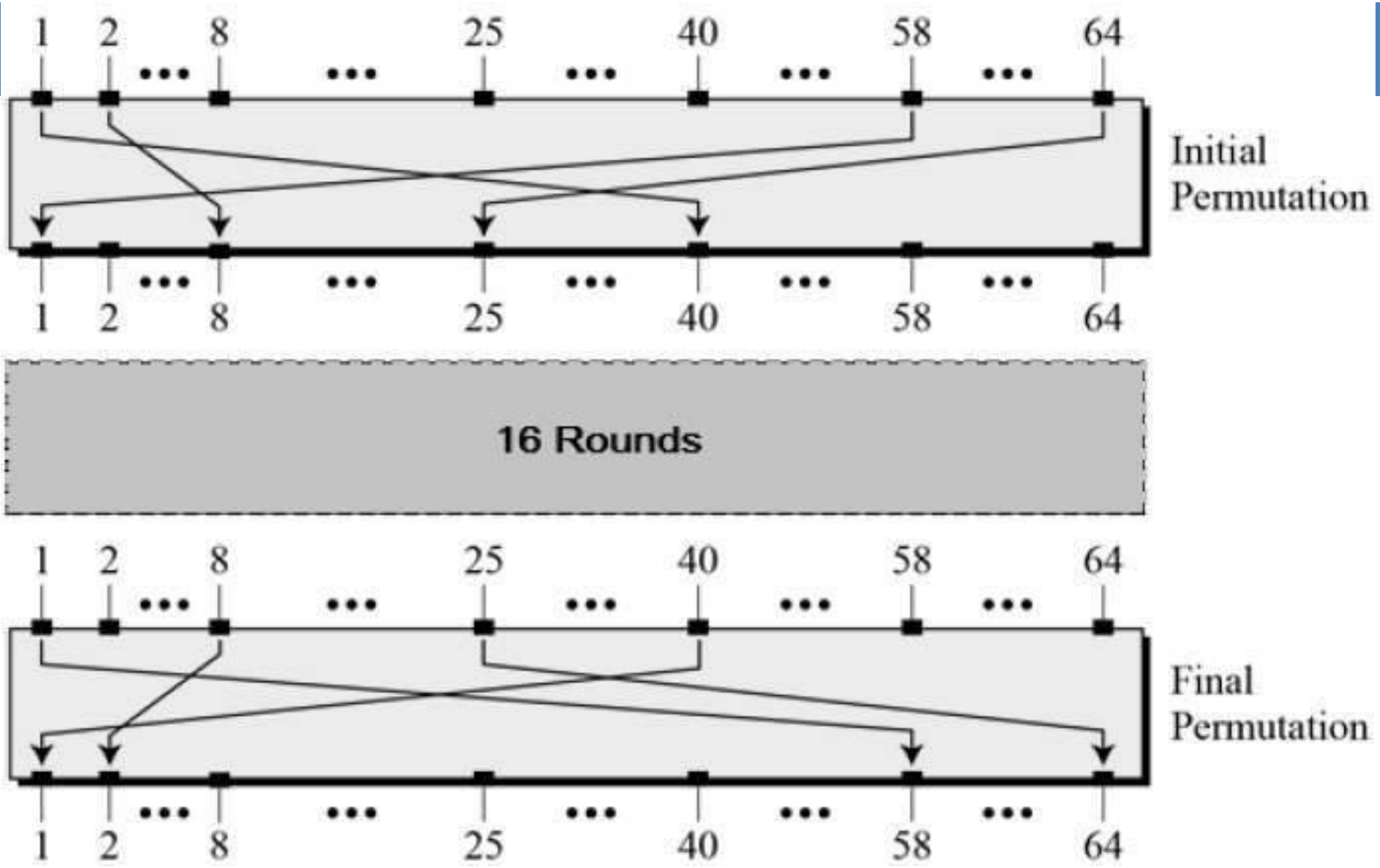

Input (64 bits)
Key (64 bits)
Initialization
Initial Permutation
Key Permutation
Left Half (32 bits)
Right Half (32 bits)
Left Half (28 bits)
Right Half (28 bits)
Binary Rotation
Binary Rotation
Round 1
Cipher Function
Permutation
Sub-key #1 (48 bits)
Binary Rotation
Binary Rotation
Round 2
Cipher Function
Permutation
Sub-key #2 (48 bits)
Binary Rotation
Binary Rotation
Round 16
Cipher Function
Permutation
Sub-key #16 (48 bits)
Finalization
Final Permutation
Output (64 bits)


Key (64 bits counting parity)
Permuted Choice 1
C0(28 bits)
D0(28 bits)
Left Shift
Left Shift
C1
D1
Permuted Choice 2
K1(48 bits)
Left Shift
Left Shift
C2
D2
Permuted Choice 2
K2(48 bits)
Left Shift
Left Shift
C3
D3
C16
D16
Permuted Choice 2
K16(48 bits)

Initial Permutation IP

- first step of the data computation
- IP reorders the input data bits
- even bits to LH half, odd bits to RH half
- quite regular in structure (easy in h/w)
- see text Table 3.2
- example:

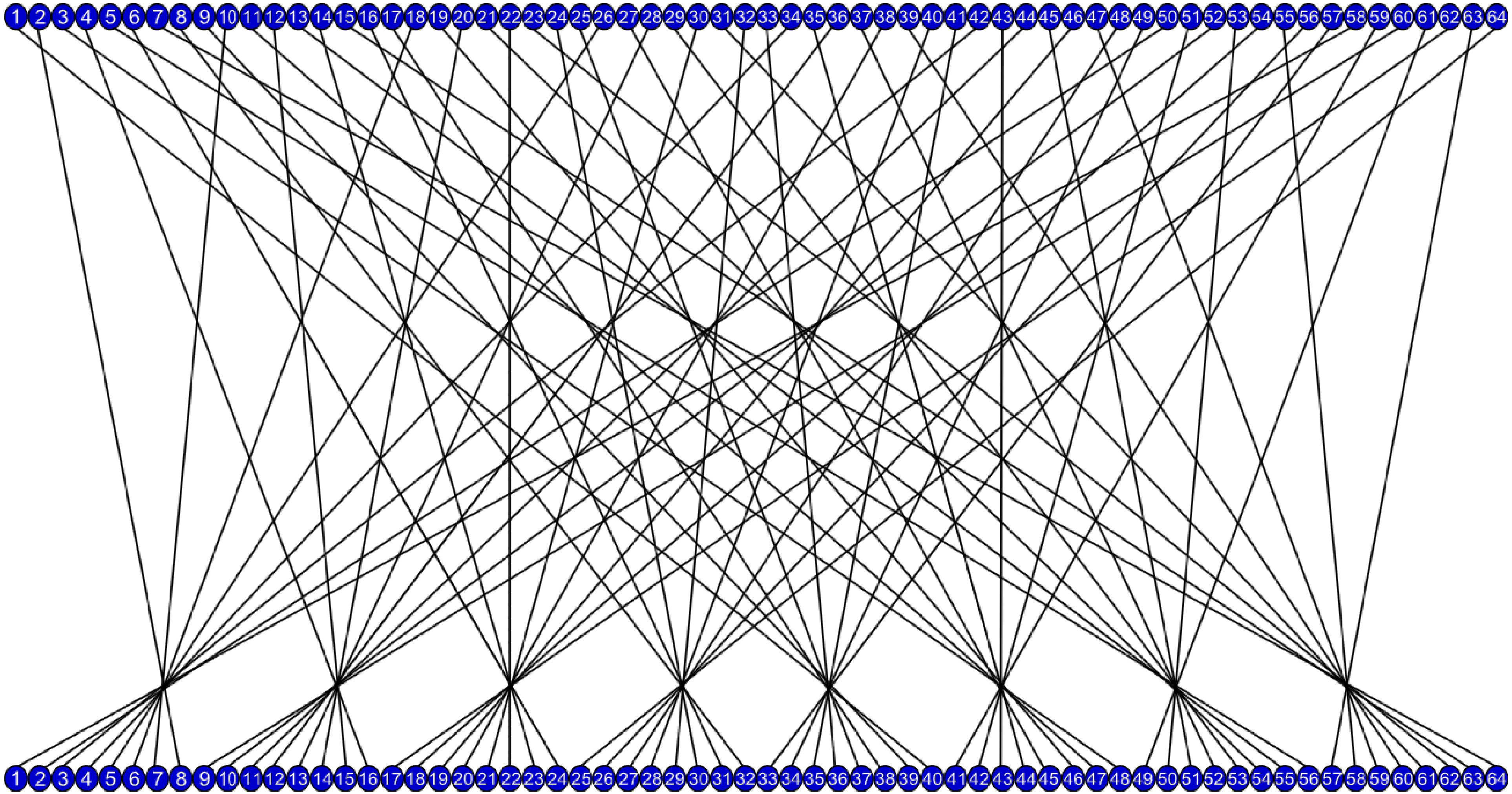
```
IP(675a6967 5e5a6b5a) = (ffb2194d 004df6fb)
```


1 2 8 25 40 58 64
Initial Permutation
1 2 8 25 40 58 64
16 Rounds
1 2 8 25 40 58 64
Final Permutation
1 2 8 25 40 58 64

Initial Permutation 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

Initial Permutation IP

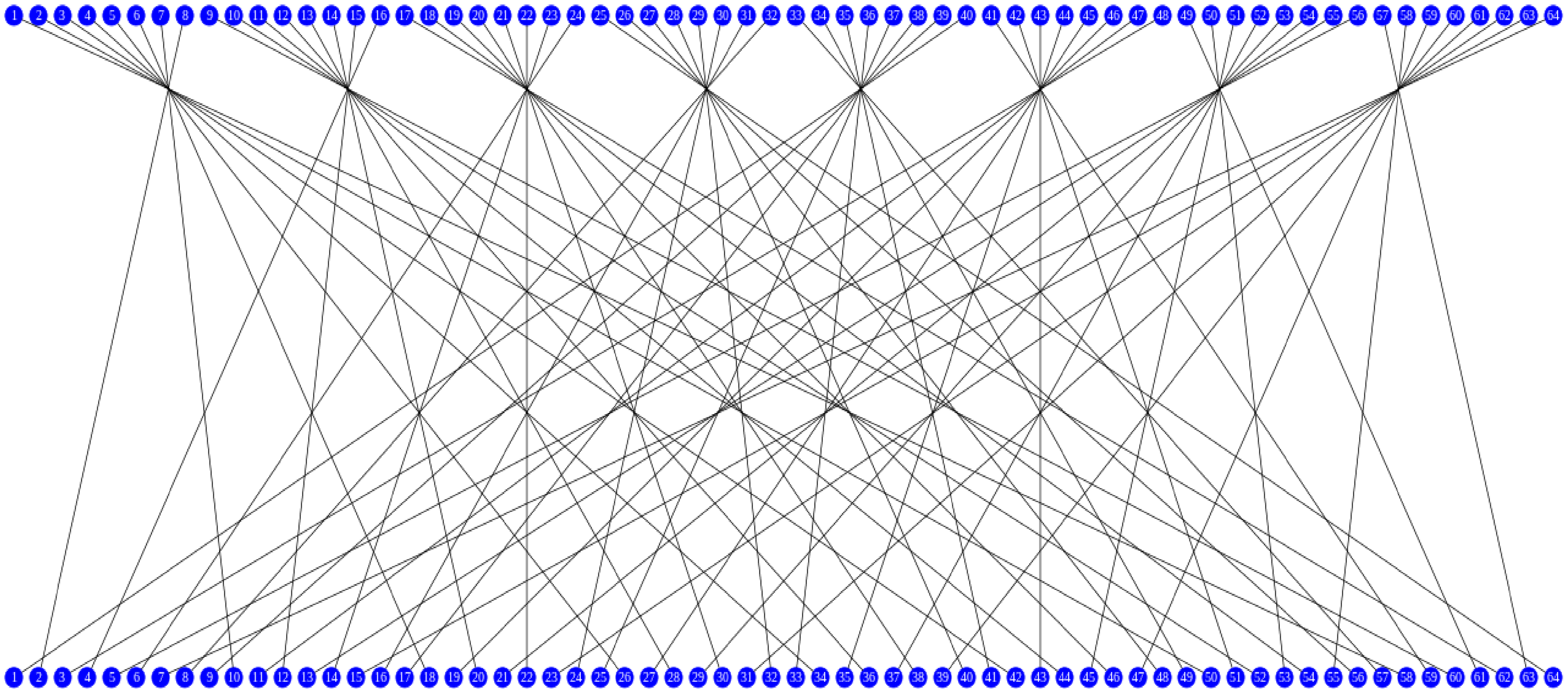


Inverse Initial Permutation

- The **Inverse Initial Permutation** is:

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

IP^{-1}


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
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

Expansion Table E

- Expands the 32 bit data to 48 bits
  - Result(i)=input(array(i))

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

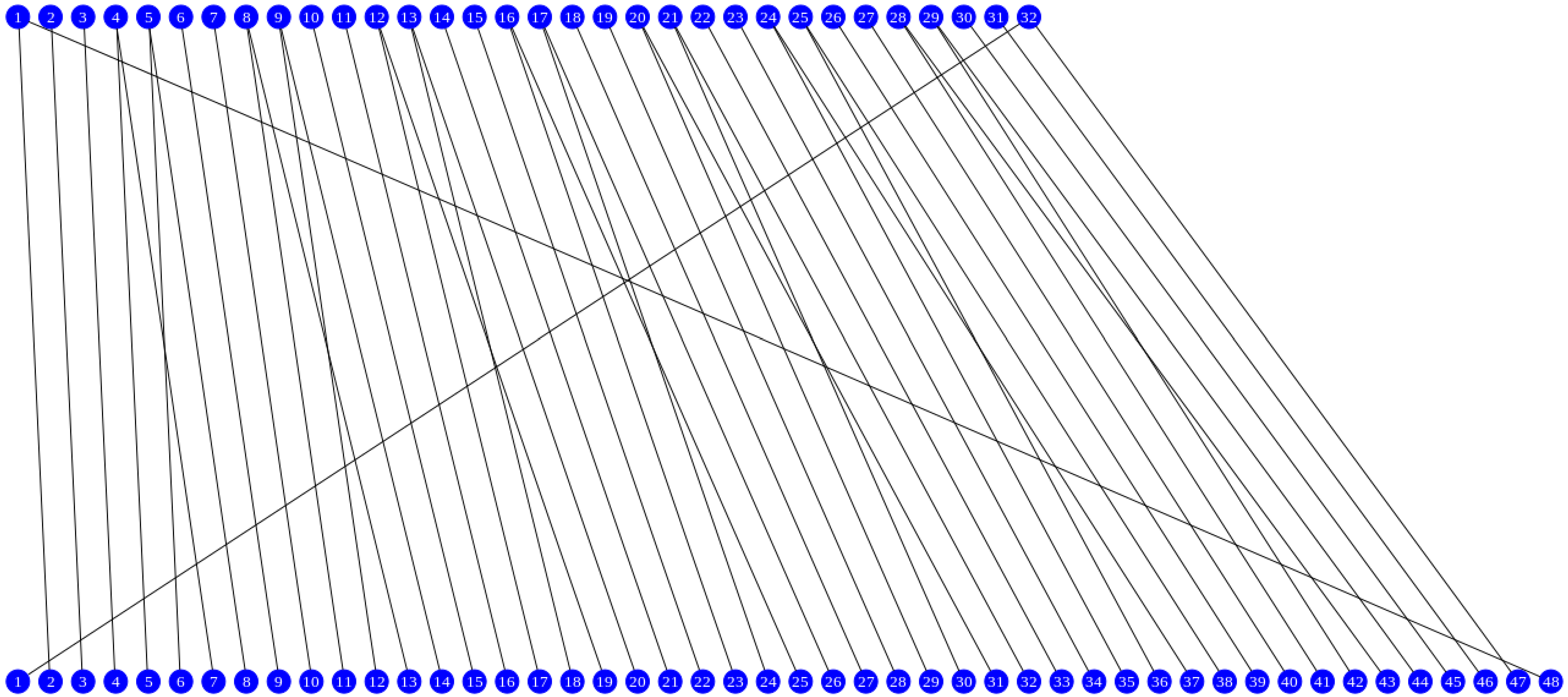
Expansion Function (E)

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

Expansion Function (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

Expansion Function (E)



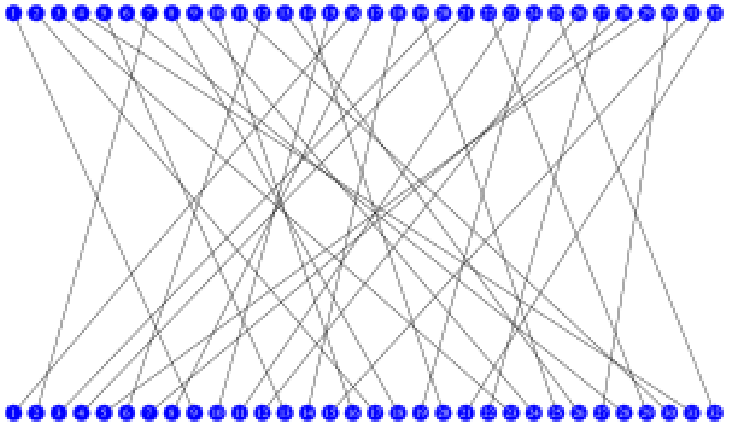
Permutation

Permutation (P) [edit]

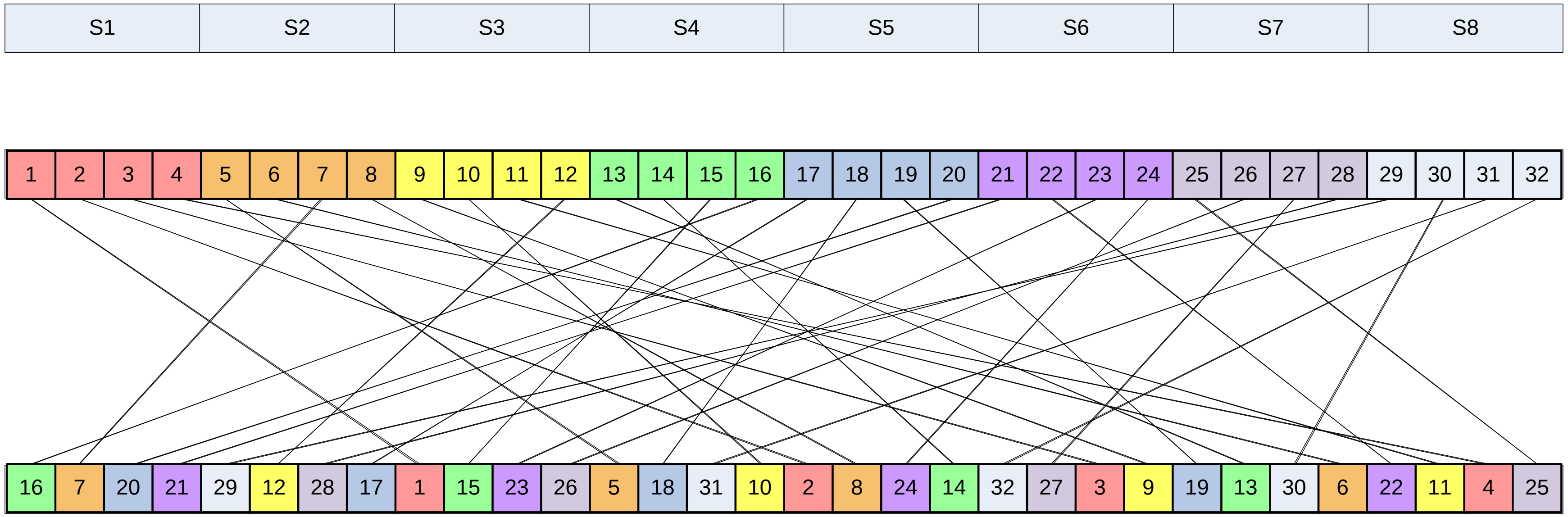
P

16	7	20	21	29	12	28	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

The P permutation shuffles the bits of a 32-bit half-block.



Permutation Box P



- **P-box applied at end of each round**
- **Increases diffusion/avalanche effect**

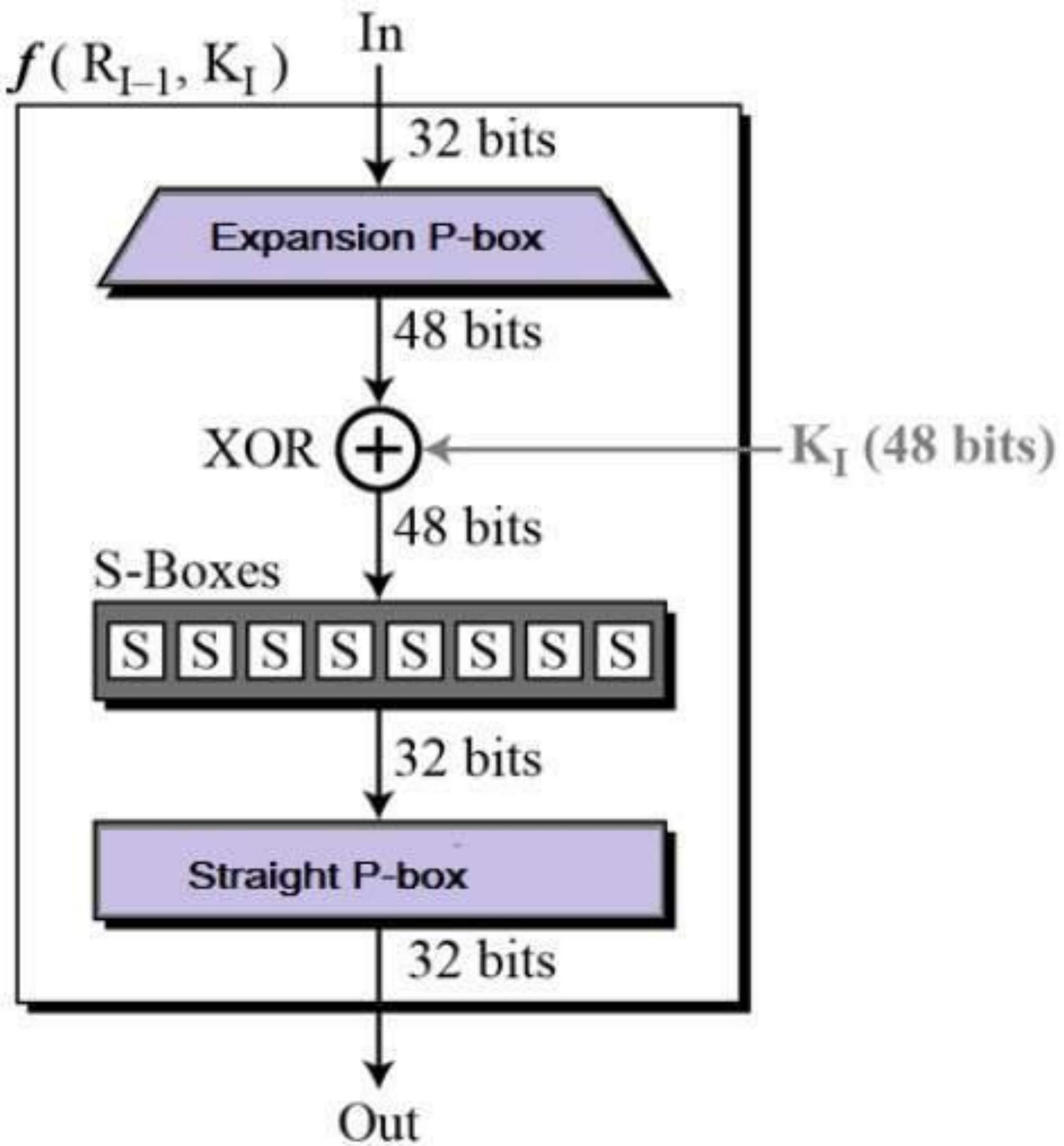
DES Round Structure

- uses two 32-bit L & R halves
- as for any **Feistel cipher** can describe as:

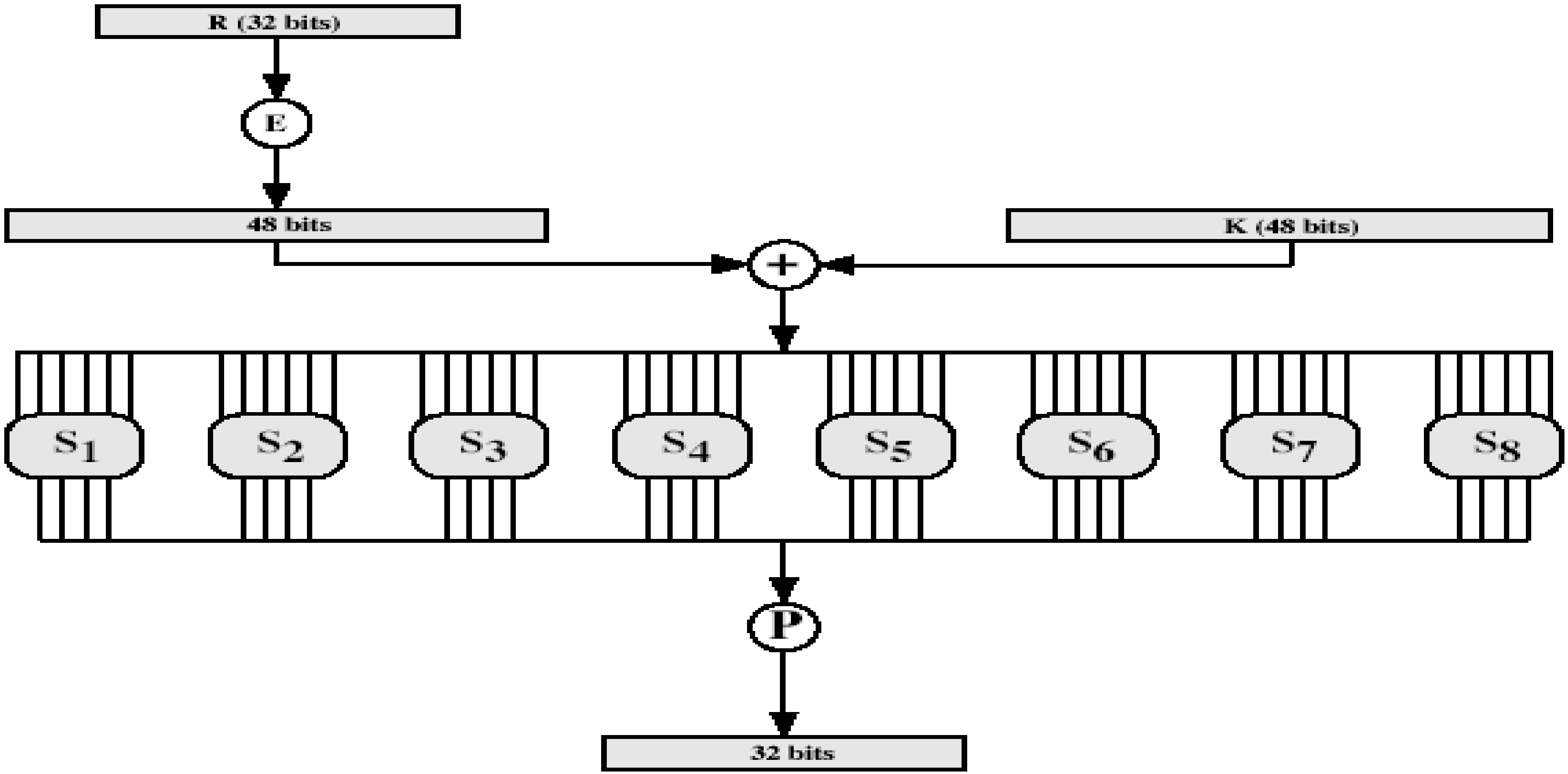
  $L_i = R_{i-1}$

  $R_i = L_{i-1} \text{ xor } F(R_{i-1}, K_i)$
- takes 32-bit R half and 48-bit subkey and:
  – expands R to 48-bits using perm E
  – adds to subkey
  – passes through 8 S-boxes to get 32-bit result
  – finally permutes this using 32-bit perm P

Round Function



DES Round Structure



S-Boxes

- S-Box is a fixed 4 by 16 array
- Given 6-bits B= $b_1b_2b_3b_4b_5b_6$,
  - Row r= b_1b_6
  - Column c= $b_2b_3b_4b_5$
  - S(B)=S(r,c) written in binary of length 4

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 rows
  - 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
```

Example

- S-Box S_1

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

S-Box Example

Input: 011011

Outer bit: 011011 => 01

Middle Bits 011011 => 1101

S_5		Middle 4 bits of input															
		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	**1101**	1110	1111
Outer bits	00	0010	1100	0100	0001	0111	1010	1011	0110	1000	0101	0011	1111	1101	0000	1110	1001
	01	1110	1011	0010	1100	0100	0111	1101	0001	0101	0000	1111	1010	0011	1001	1000	0110
	10	0100	0010	0001	1011	1010	1101	0111	1000	1111	1001	1100	0101	0110	0011	0000	1110
	11	1011	1000	1100	0111	0001	1110	0010	1101	0110	1111	0000	1001	1010	0100	0101	0011

For example, an input "**011011**" has outer bits "**01**" and inner bits "1101"; the corresponding output would be "1001".

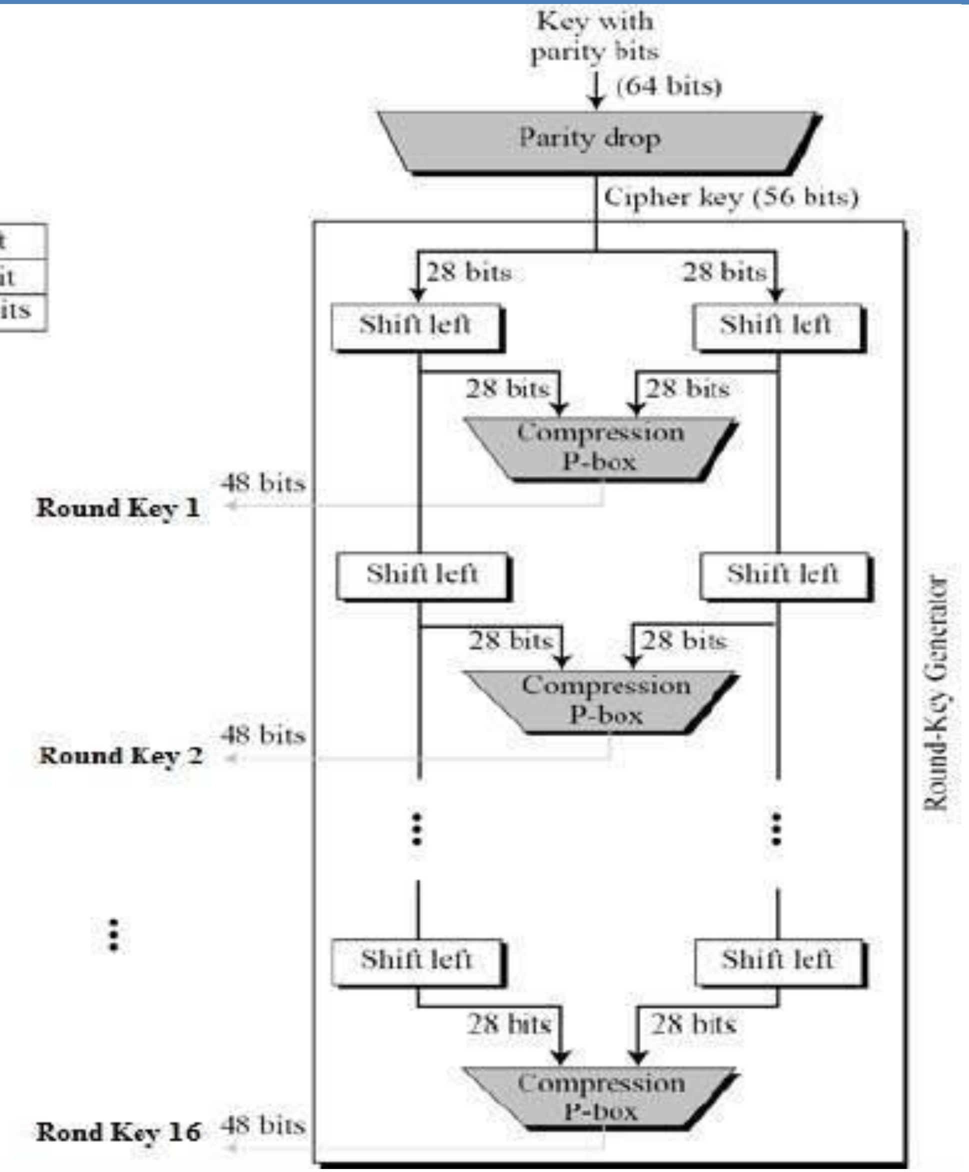
DES Key Schedule

- forms subkeys used in each round
- consists of:
  - initial permutation of the key (PC1) which selects 56-bits in two 28-bit halves
  - 16 stages consisting of:
    - selecting 24-bits from each half
    - permuting them by PC2 for use in function f,
    - rotating **each half** separately either 1 or 2 places depending on the **key rotation schedule** K

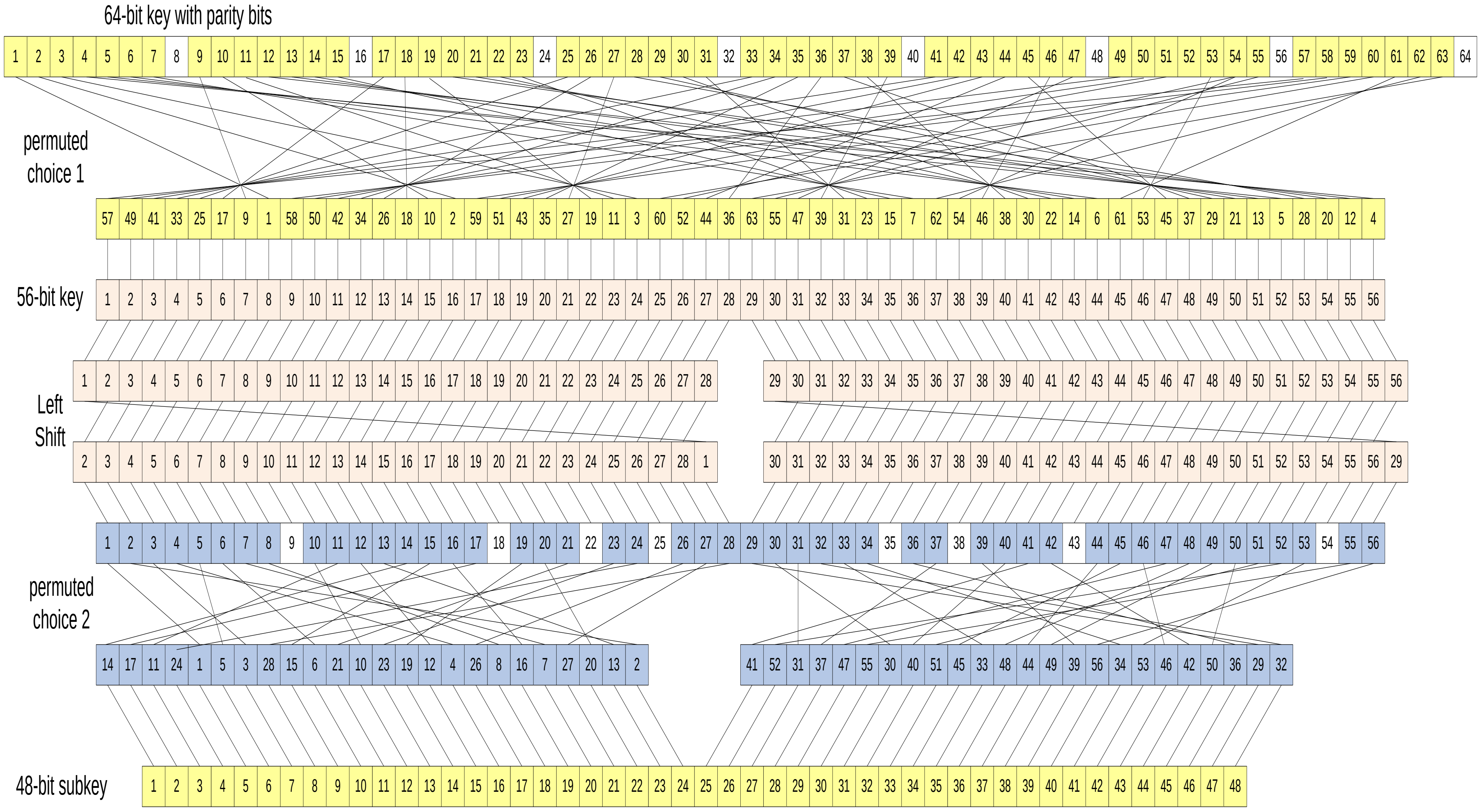
Key Generation

Shifting

Rounds	Shift
1, 2, 9, 16	one bit
Others	two bits



DES Key Schedule



Key 64 to 56

Every eighth bit is ignored and produces 56 bits.

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

Key 56

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

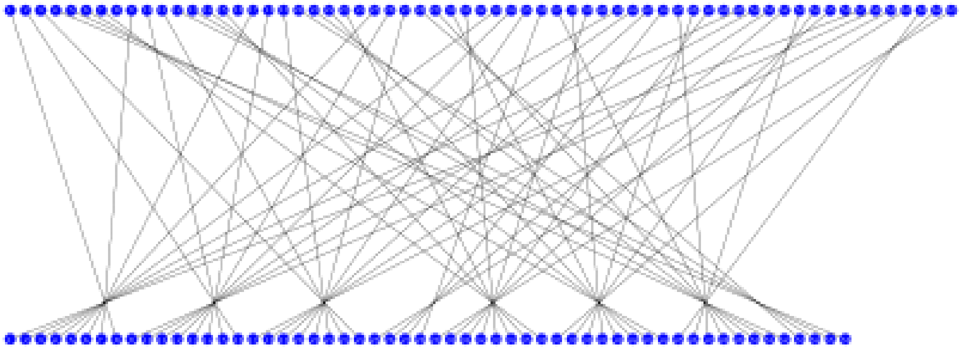
Permutation Choice one (PC-1)

- **56 bits pass** through a **permutation Choice one (PC-1)** and displays as follows:

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	21	4

PC-1

Permuted choice 1 (PC-1) [edit]



PC-1

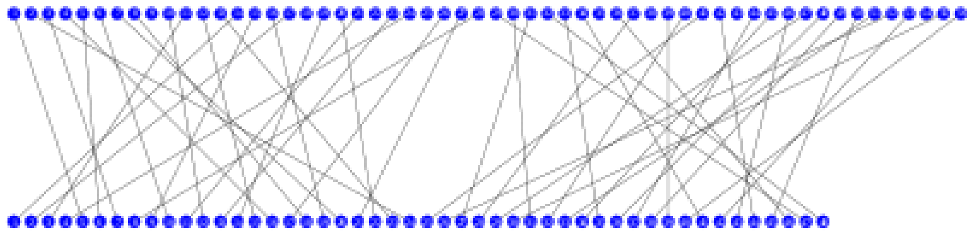
Left							Right						
57	49	41	33	25	17	9	63	55	47	39	31	23	15
1	58	50	42	34	26	18	7	62	54	46	38	30	22
10	2	59	51	43	35	27	14	6	61	53	45	37	29
19	11	3	60	52	44	36	21	13	5	28	20	12	4

PC-2

Permuted choice 2 (PC-2) [edit]

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 Decryption

- decrypt must unwind steps of data computation
- with Feistel design, do encryption steps again
- using **subkeys in reverse order (SK16 … SK1)**
- note that 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

DES Example

Round	K_i	L_i	R_i
IP		5a005a00	3cf03c0f
1	1e030f03080d2930	3cf03c0f	bad22845
2	0a31293432242318	bad22845	99e9b723
3	23072318201d0c1d	99e9b723	0bae3b9e
4	05261d3824311a20	0bae3b9e	42415649
5	3325340136002c25	42415649	18b3fa41
6	123a2d0d04262a1c	18b3fa41	9616fe23
7	021f120b1c130611	9616fe23	67117cf2
8	1c10372a2832002b	67117cf2	c11bfc09
9	04292a380c341f03	c11bfc09	887fbc6c
10	2703212607280403	887fbc6c	600f7e8b
11	2826390c31261504	600f7e8b	f596506e
12	12071c241a0a0f08	f596506e	738538b8
13	300935393c0d100b	738538b8	c6a62c4e
14	311e09231321182a	c6a62c4e	56b0bd75
15	283d3e0227072528	56b0bd75	75e8fd8f
16	2921080b13143025	75e8fd8f	25896490
IP^{-1}		da02ce3a	89ecac3b

Avalanche Effect

- key desirable property of encryption alg
- 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** exhibits **strong avalanche**

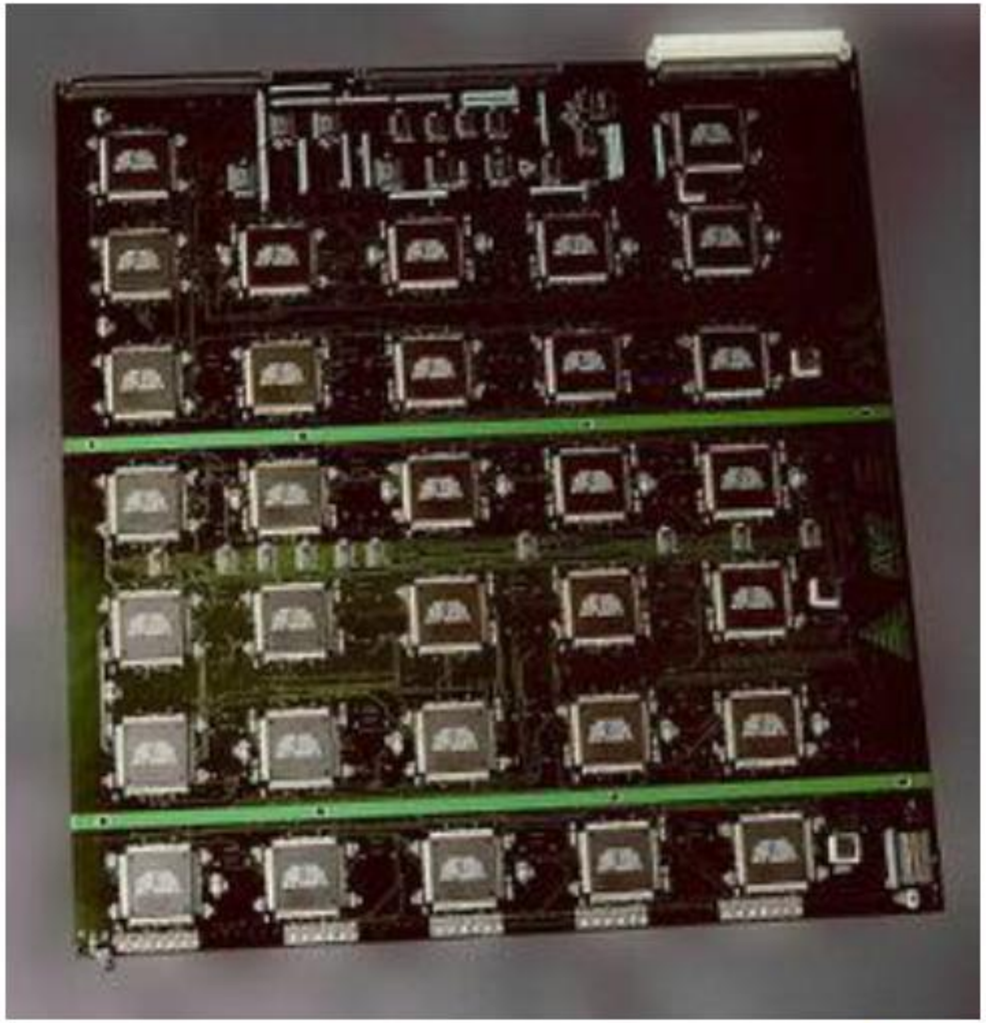
Avalanche in DES

Round		δ	Round		δ
	02468aceeca86420 12468aceeca86420	1	9	c11bfc09887fbc6c 99f911532eed7d94	32
1	3cf03c0fbad22845 3cf03c0fbad32845	1	10	887fbc6c600f7e8b 2eed7d94d0f23094	34
2	bad2284599e9b723 bad3284539a9b7a3	5	11	600f7e8bf596506e d0f23094455da9c4	37
3	99e9b7230bae3b9e 39a9b7a3171cb8b3	18	12	f596506e738538b8 455da9c47f6e3cf3	31
4	0bae3b9e42415649 171cb8b3ccaca55e	34	13	738538b8c6a62c4e 7f6e3cf34bc1a8d9	29
5	4241564918b3fa41 ccaca55ed16c3653	37	14	c6a62c4e56b0bd75 4bc1a8d91e07d409	33
6	18b3fa419616fe23 d16c3653cf402c68	33	15	56b0bd7575e8fd8f 1e07d4091ce2e6dc	31
7	9616fe2367117cf2 cf402c682b2cefbc	32	16	75e8fd8f25896490 1ce2e6dc365e5f59	32
8	67117cf2c11bfc09 2b2cefbc99f91153	33	IP^{-1}	da02ce3a89ecac3b 057cde97d7683f2a	32

Strength of DES – Key Size

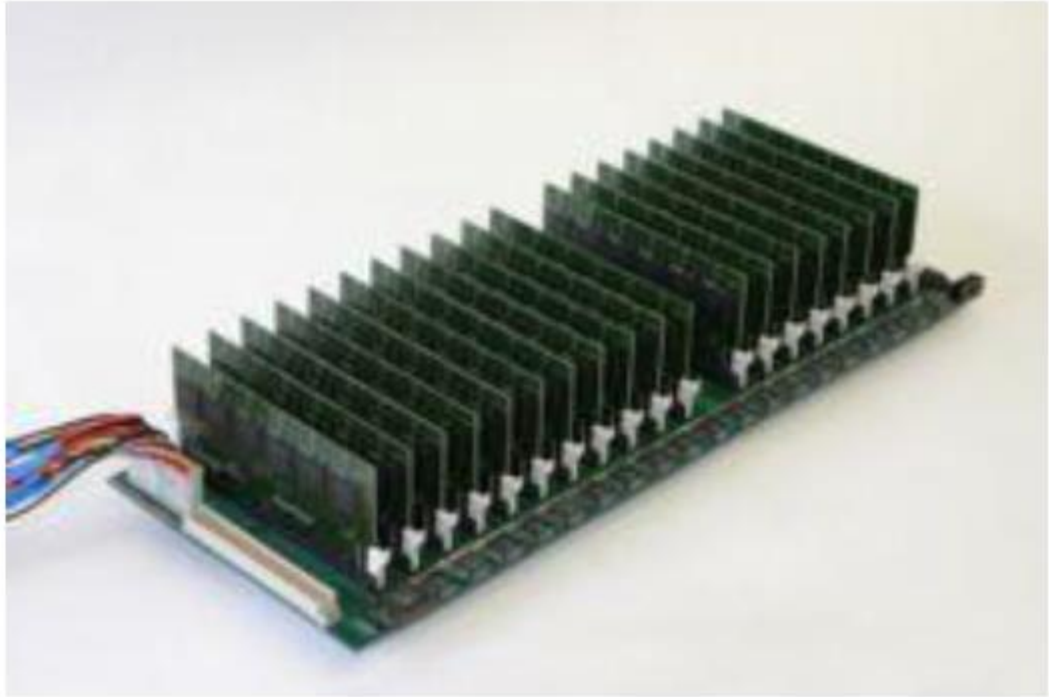
- **56-bit** keys have **2^{56} = 7.2 x 10^{16} values**
- brute force search looks hard
- recent advances have shown is possible
  - in 1997 on Internet in a few months
  - in 1998 on dedicated h/w the Electronic Frontier Foundation (EFF) in a few days
  - in 1999 above combined in 22hrs!
- still must be able to recognize plaintext
- now considering alternatives to DES

DES Attacks



1998:
The EFF's US$250,000
DES cracking machine
contained 1,536 custom chips
and could brute force a DES key in a
matter of days —
the photo shows a DES Cracker
circuit board fitted
with several Deep Crack chips.

DES Attacks:



The COPACOBANA machine, built for US$10,000 by the Universities of Bochum and Kiel, contains 120 low-cost FPGAs and can perform an exhaustive key search on DES in 9 days on average. The photo shows the backplane of the machine with the FPGAs

Strength of DES – Timing Attacks

- attacks actual implementation of cipher
- use knowledge of consequences of implementation to derive knowledge of some/all subkey bits
- specifically use fact that calculations can take varying times depending on the value of the inputs to it
- particularly problematic on smartcards

Strength of DES – Analytic Attacks

- now 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**

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 1990**
- 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 values of the key
- Differential Cryptanalysis **compares two related pairs** of encryptions

Differential Cryptanalysis Compares Pairs of Encryptions

- with a known difference in the input
- searching for a known difference in output
- when **same subkeys** are used

$$\Delta m_{i+1} = m_{i+1} \oplus m'_{i+1}$$

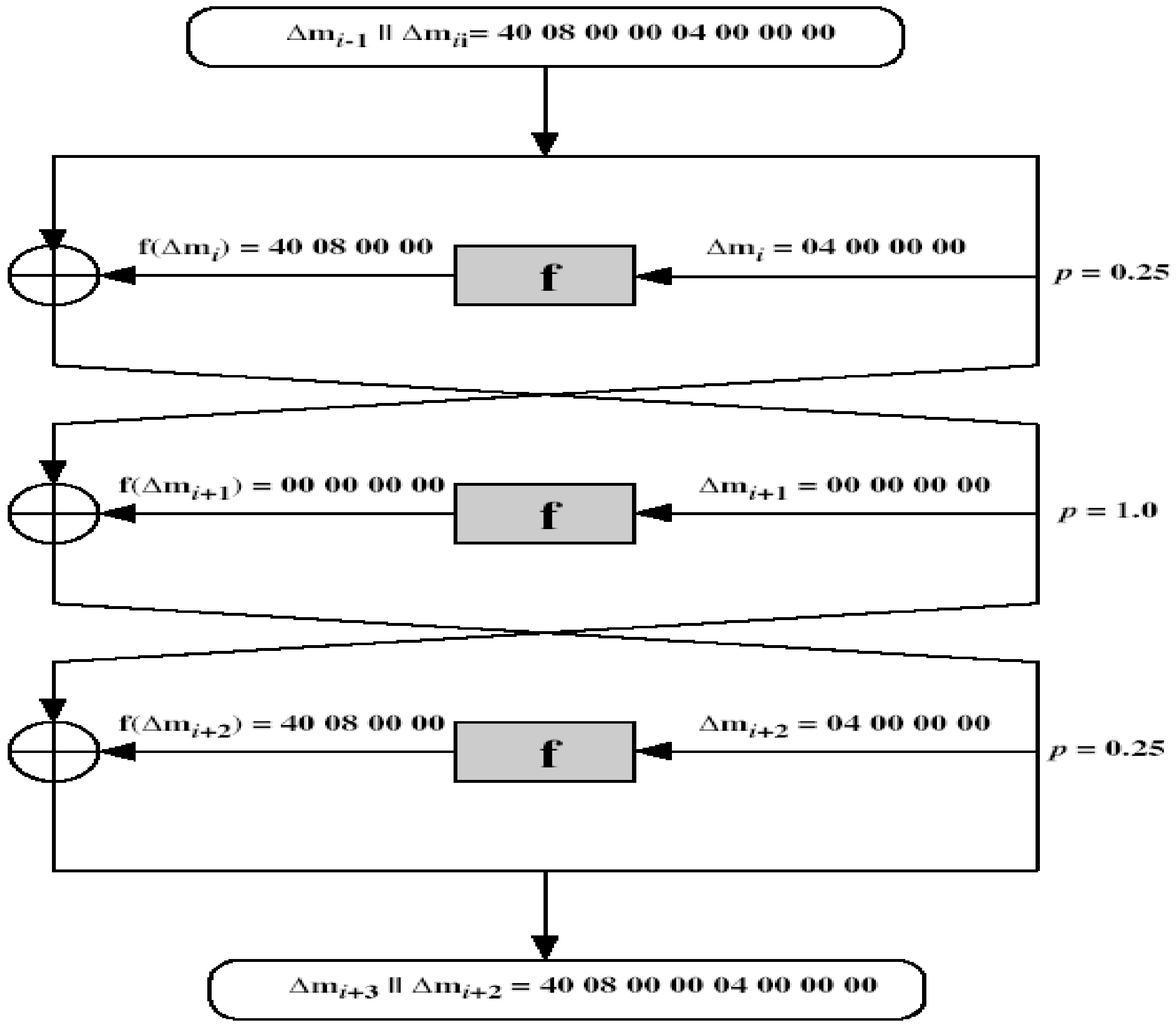
$$= \left[m_{i-1} \oplus f(m_i, K_i)\right] \oplus \left[m'_{i-1} \oplus f(m'_i, K_i)\right]$$

$$= \Delta m_{i-1} \oplus \left[f(m_i, K_i) \oplus f(m'_i, K_i)\right]$$

Differential Cryptanalysis

- have some input difference giving some output difference with probability p
- if find instances of some higher probability input / output difference pairs occurring
- can infer subkey that was used in round
- then must iterate process over many rounds (with decreasing probabilities)

Differential Cryptanalysis



Differential Cryptanalysis

- perform attack by repeatedly encrypting plaintext pairs with known input XOR until obtain desired output XOR
- when found
  - if intermediate rounds match required XOR have a **right pair**
  - if not then have a **wrong pair**, relative ratio is S/N for attack
- can then deduce keys values for the rounds
  - right pairs suggest same key bits
  - wrong pairs give random values
- for large numbers of rounds, **probability is so low** that more pairs are required than exist with **64-bit inputs**
- Biham and **Shamir** have shown how a 13-round iterated characteristic can break the full 16-round DES

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^{47} known plaintexts, still in practise infeasible

Linear Cryptanalysis

- find linear approximations with **prob p != ½**

  P[i1,i2,...,ia](+)C[j1,j2,...,jb] = K[k1,k2,...,kc]

  where i_a, j_b, k_c are bit locations in **P,C,K**
- gives linear equation for key bits
- get one key bit using max likelihood alg
- using a large number of trial encryptions
- effectiveness given by: **|p–½|**

Block Cipher Design Principles

- basic principles still like Feistel in 1970's
- **number of rounds**
  - more is better, exhaustive search best attack
- **function f:**
  - provides "confusion", is nonlinear, avalanche
- **key schedule**
  - complex subkey creation, key avalanche

Modes of Operation

- block ciphers encrypt fixed size blocks
- eg. DES encrypts **64-bit blocks, with 56-bit key**
- need way to use in practise, given usually have arbitrary amount of information to encrypt
- four were defined for DES in ANSI standard **ANSI X3.106-1983 Modes of Use**
- subsequently now have **5 for DES and AES**
- have two types of modes:
  1. block and (two block modes)
  2. **stream modes** (three stream modes)

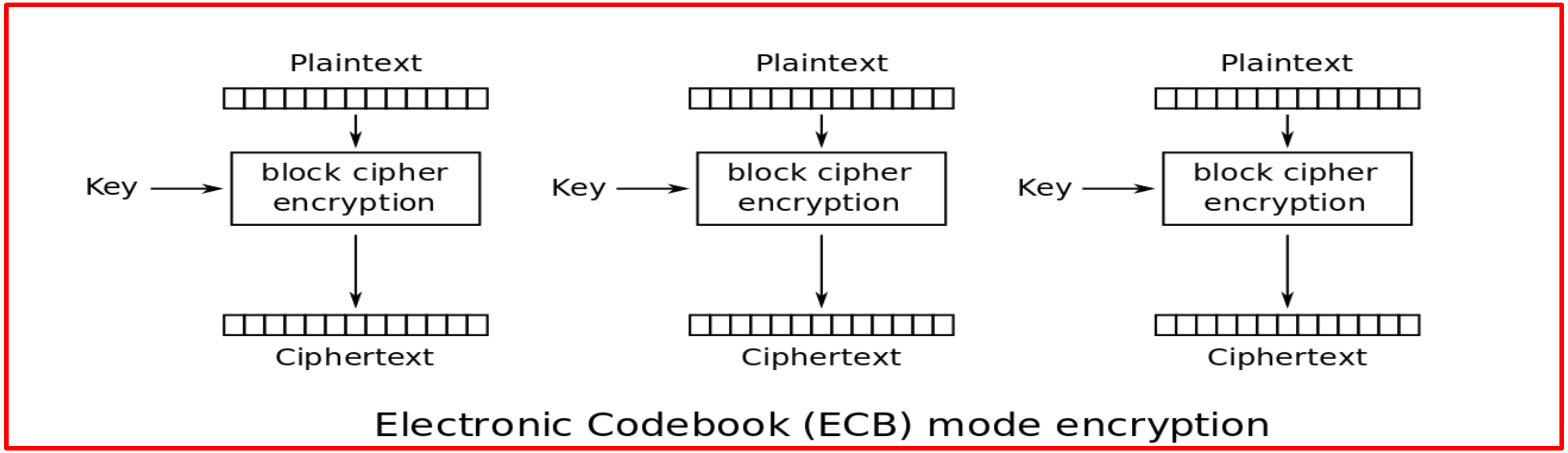
Electronic Codebook Book (ECB)

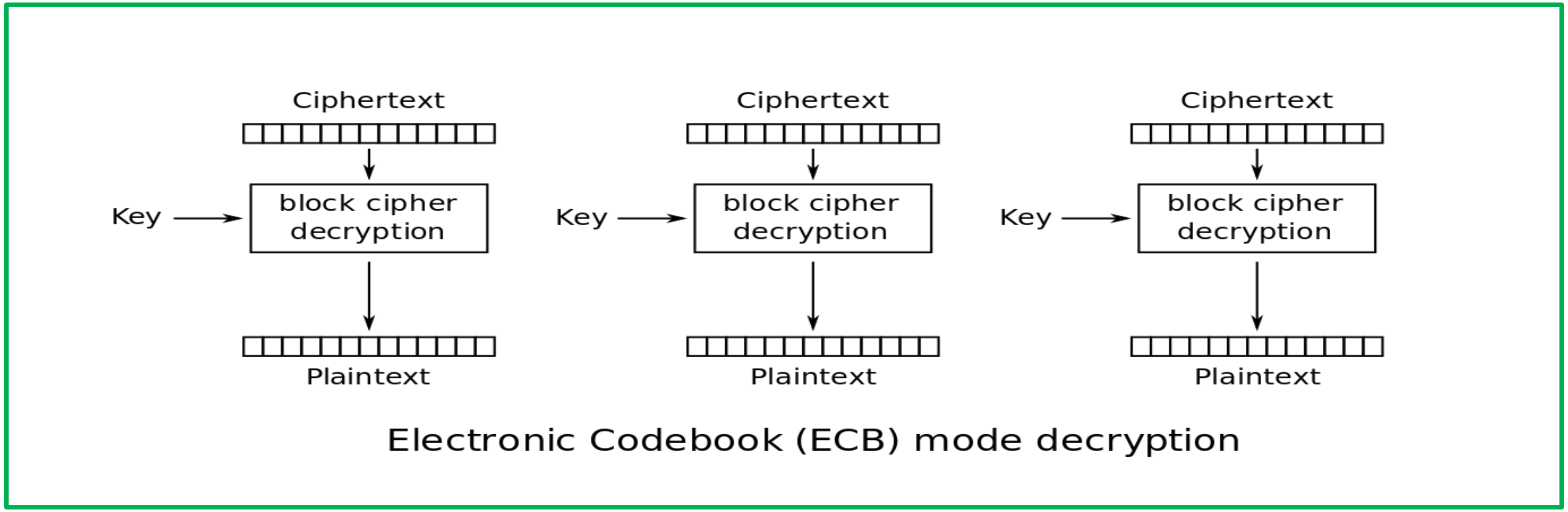
- message is broken into independent blocks which are encrypted
- each block is a value which is substituted, like a **codebook**, hence name
- **each block** is encoded **independently** of the other blocks

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

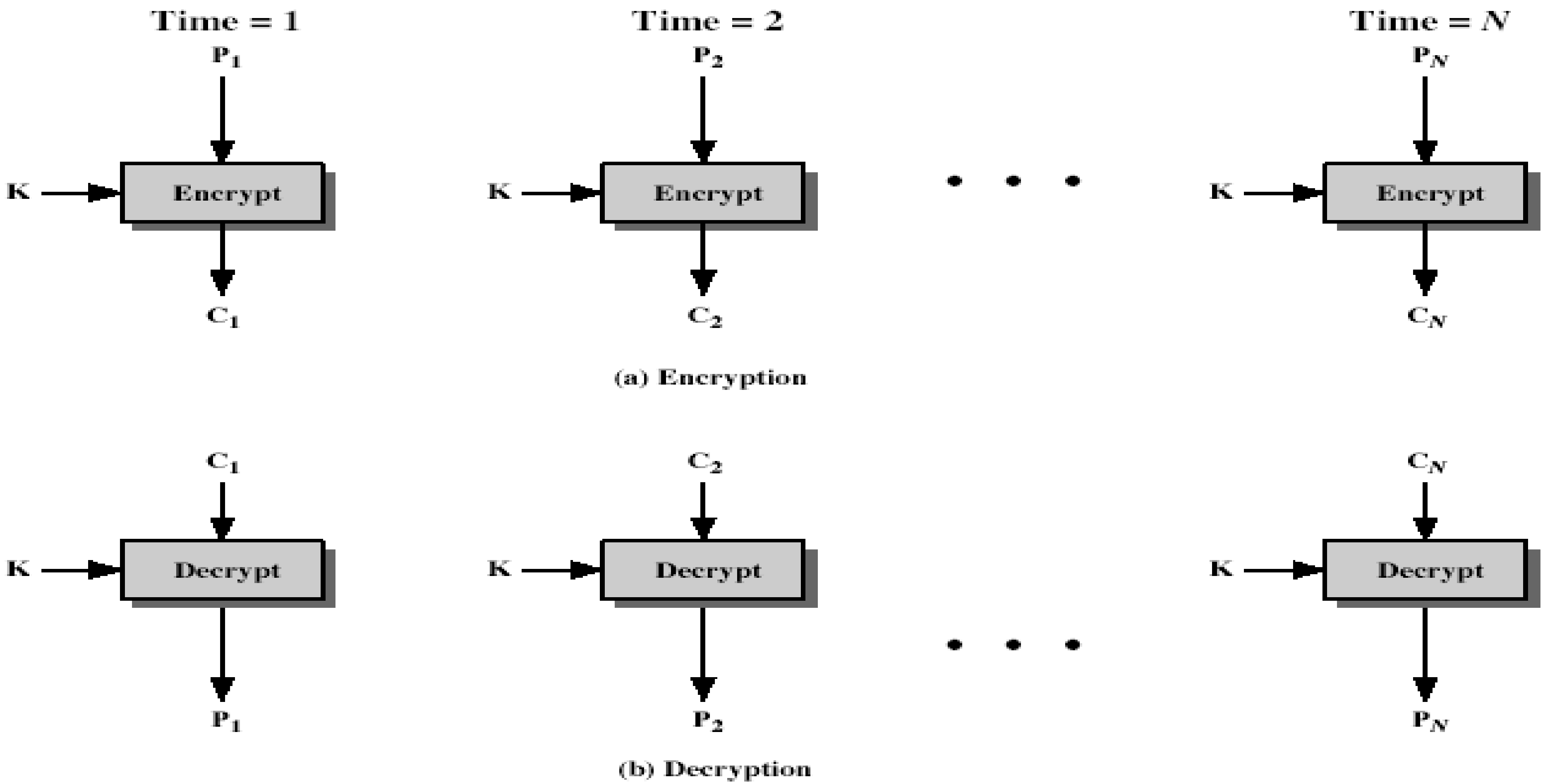
- *uses:* secure transmission of single values

Electronic Codebook Book (ECB)





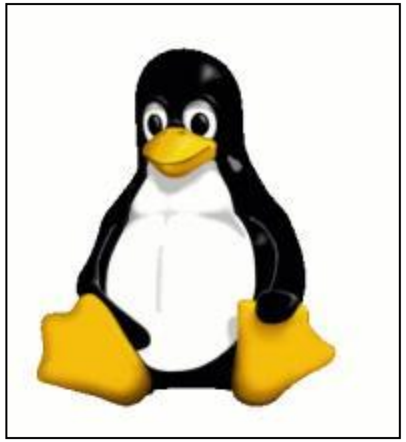
Electronic Codebook Book (ECB)

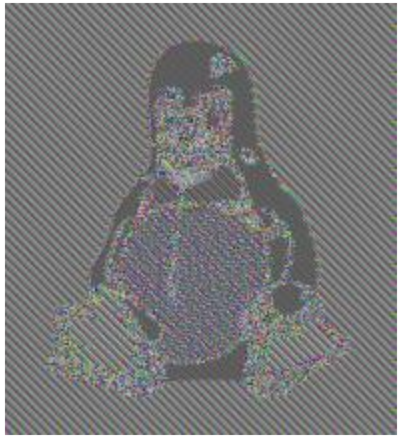


Advantages and Limitations of ECB

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

Electronic Codebook Book (ECB)





- Original image

Encrypted using ECB mode

Advantages and Limitations of ECB

ECB	
Electronic Codebook	
Encryption parallelizable:	Yes
Decryption parallelizable:	Yes
Random read access:	Yes

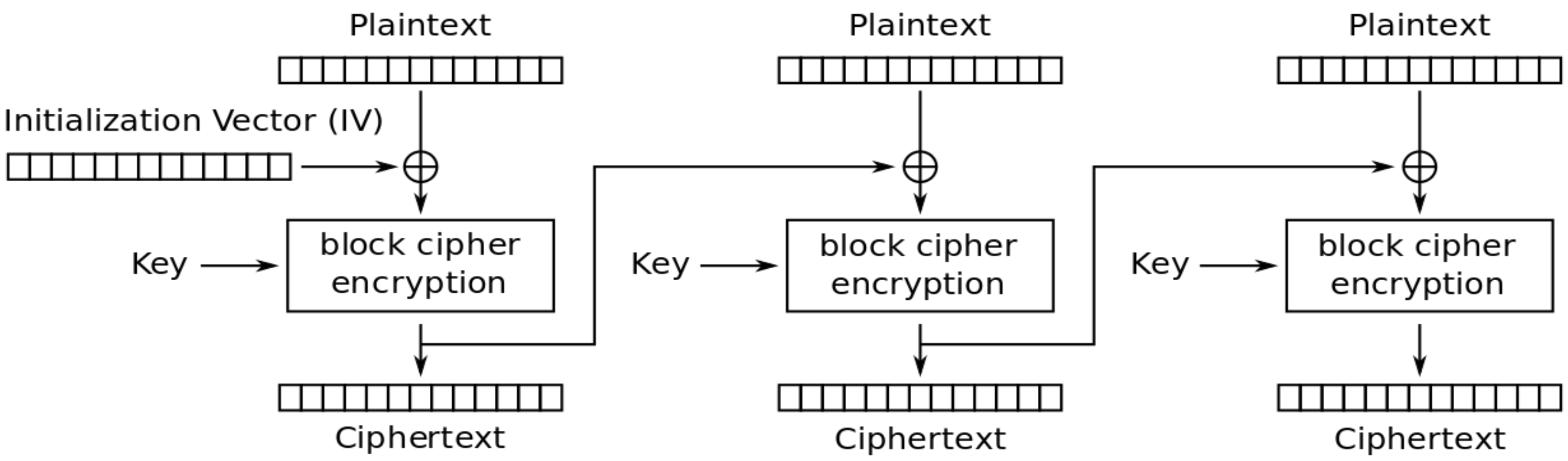
Cipher Block Chaining (CBC)

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

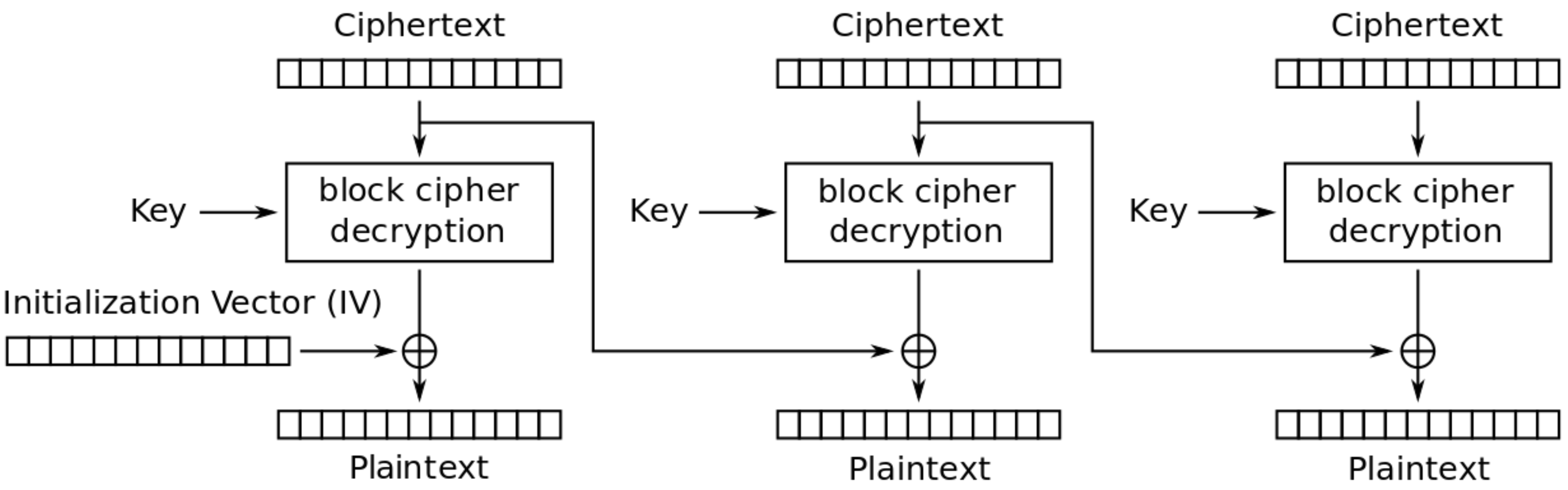
$$C_i = DES_{K1}(P_i \text{ XOR } C_{i-1})$$

$$C_{-1} = IV$$

- uses: **bulk data encryption, authentication**

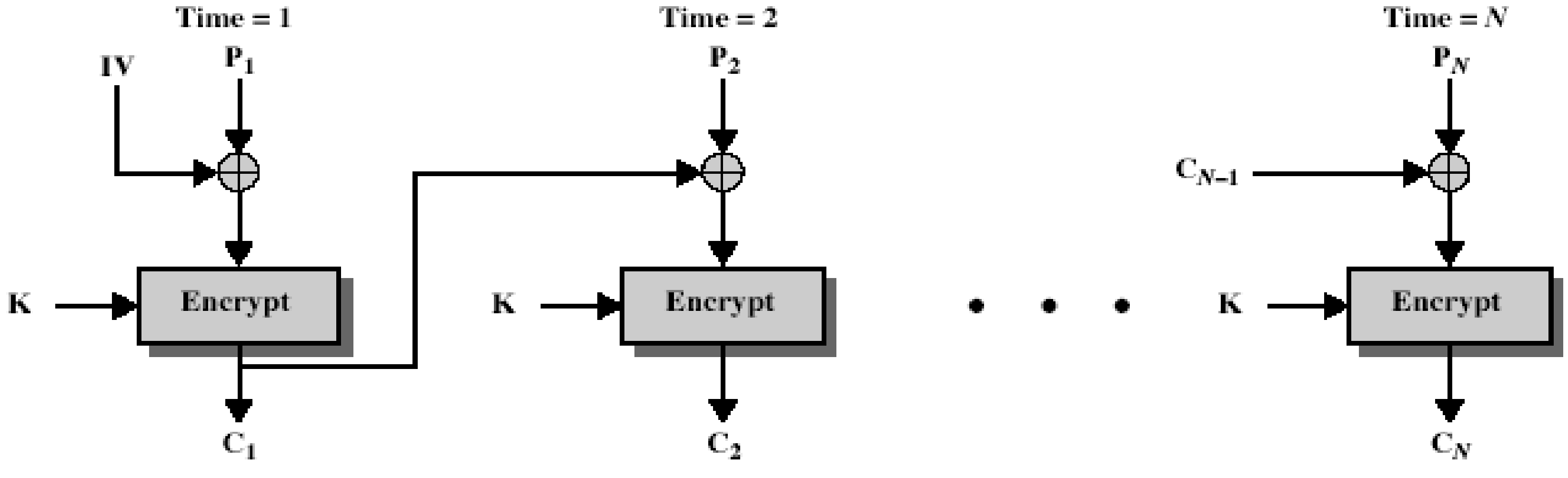


Cipher Block Chaining (CBC) mode encryption

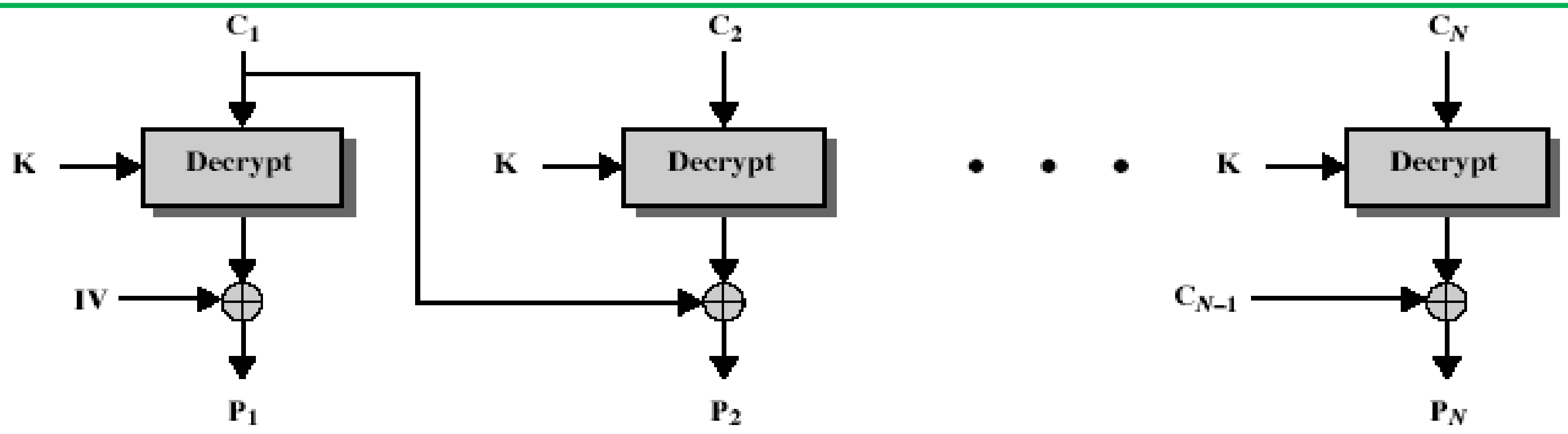


Cipher Block Chaining (CBC) mode decryption

Cipher Block Chaining (CBC)



(a) Encryption

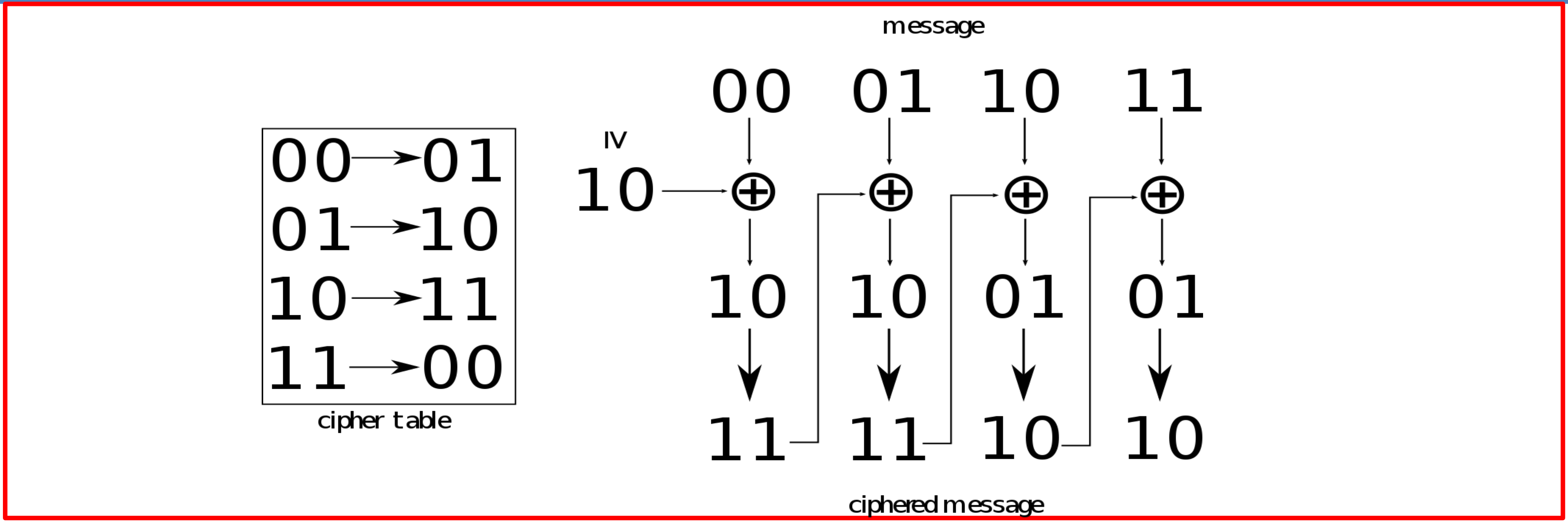


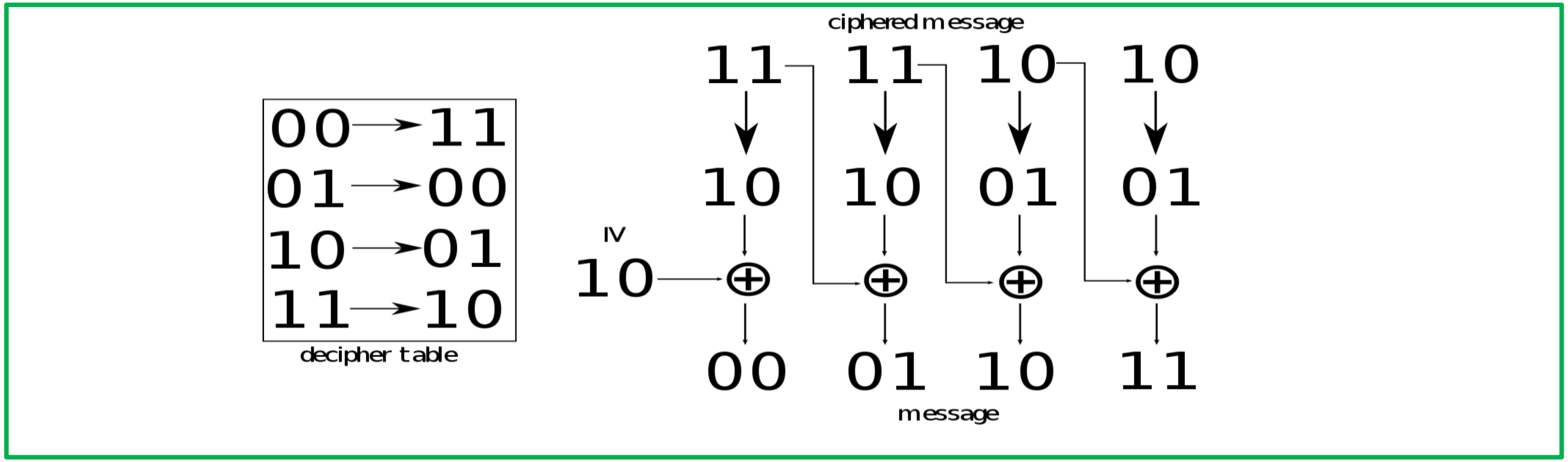
(b) Decryption

गृह मंत्रालय
MINISTRY OF
HOME AFFAIRS
सत्यमेव जयते

राष्ट्रीय न्यायिक विज्ञान विश्वविद्यालय
National Forensic Sciences University

Cipher Block Chaining (CBC) Example





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

- each ciphertext block **depends on all message blocks**
- thus a change in the message affects all ciphertext blocks after the change as well as the original block
- need **Initial Value** (IV) known to sender & receiver
  - however if IV is sent in the clear, an attacker can change bits of the first block, and change IV to compensate
  - hence either IV must be a fixed value (as in EFTPOS) or it must be sent encrypted in ECB mode before rest of message
- at end of message, handle possible last short block
  - by padding either with known non-data value (eg nulls)
  - or pad last block with count of pad size
    - eg. [b1 b2 b3 0 0 0 0 5] <- 3 data bytes, then 5 bytes pad+count

Advantages and Limitations of CBC

CBC	
Cipher Block Chaining	
Encryption parallelizable:	No
Decryption parallelizable:	Yes
Random read access:	Yes

Stream Modes of Operation

- block modes encrypt entire block
- may need to operate on smaller units
  - real time data
- convert block cipher into stream cipher
  1) cipher feedback (CFB) mode
  2) output feedback (OFB) mode
  3) counter (CTR) mode
- use block cipher as some form of **pseudo-random number** generator

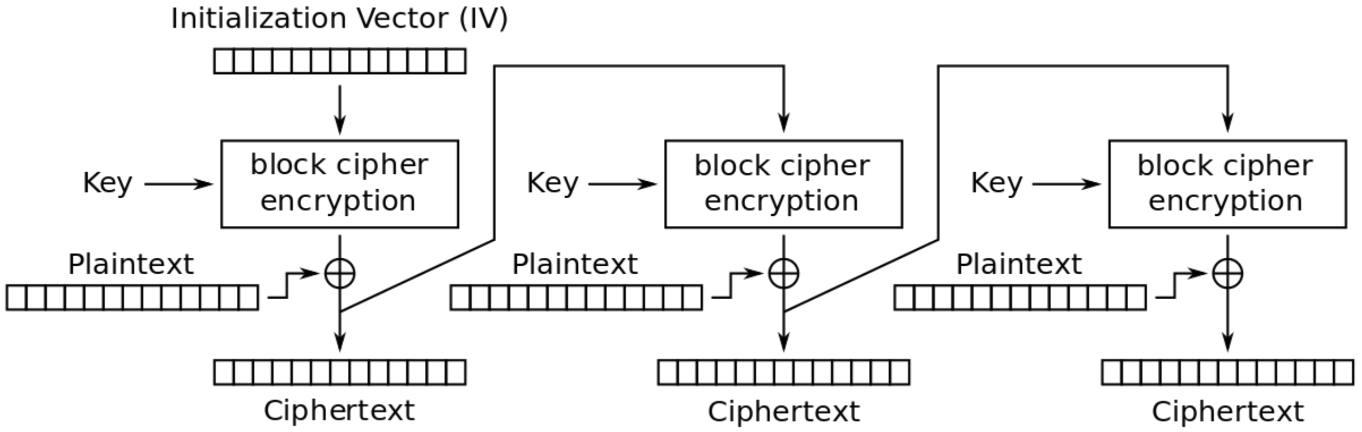
Cipher FeedBack (CFB)

- 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 or 64 or whatever) to be feed back
  - denoted CFB-1, CFB-8, CFB-64 etc
- is most efficient to **use all 64 bits (CFB-64)**

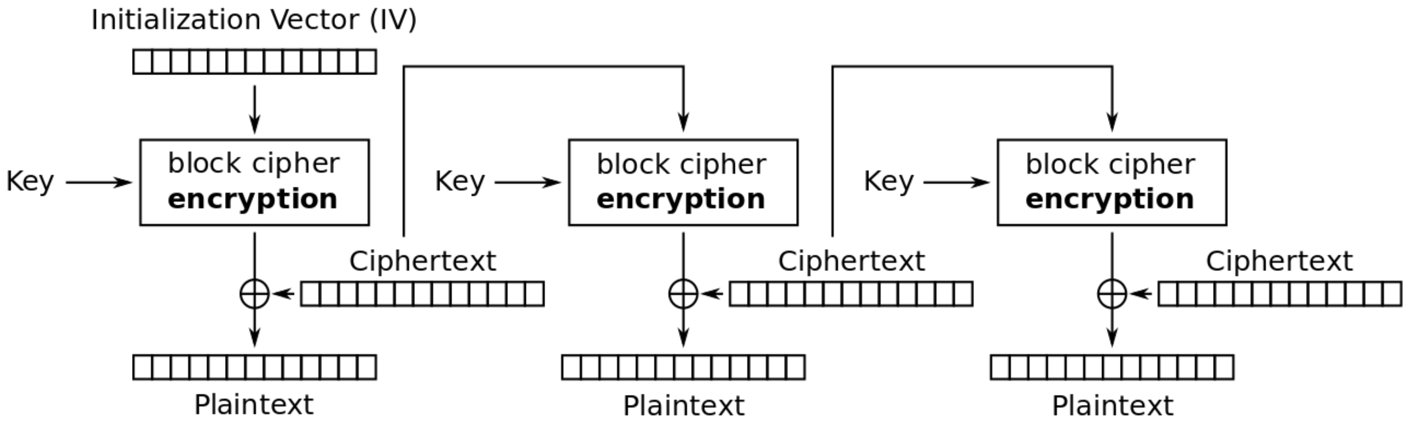
  $C_i = P_i \text{ XOR } DES_{K1}(C_{i-1})$

  $C_{-1} = IV$
- **uses:** stream data encryption, authentication

Cipher FeedBack (CFB)

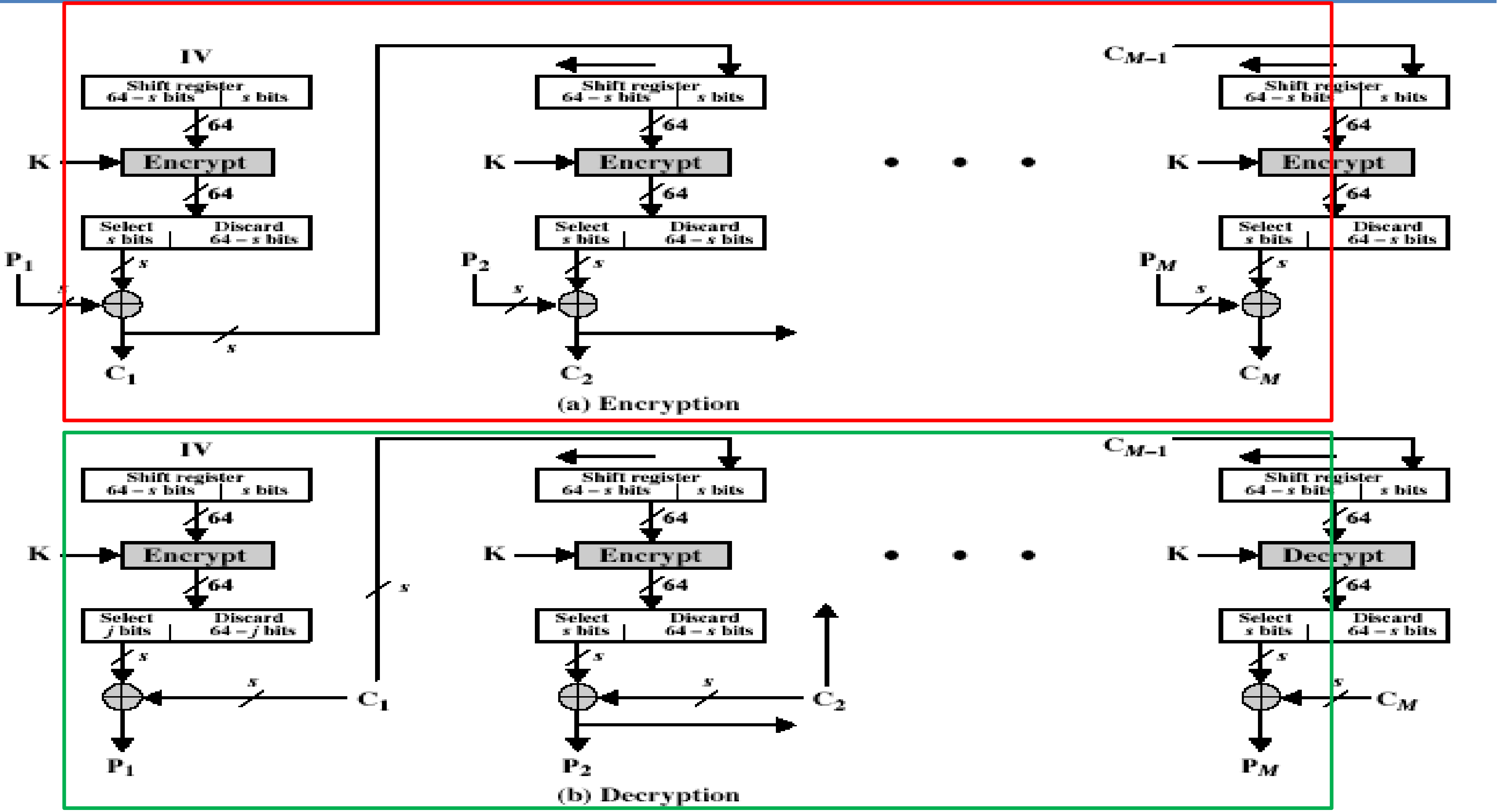


Cipher Feedback (CFB) mode encryption



Cipher Feedback (CFB) mode decryption

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
- **errors propogate for several blocks after the error**

Advantages and Limitations of CFB

CFB	
Cipher Feedback	
Encryption parallelizable:	No
Decryption parallelizable:	Yes
Random read access:	Yes

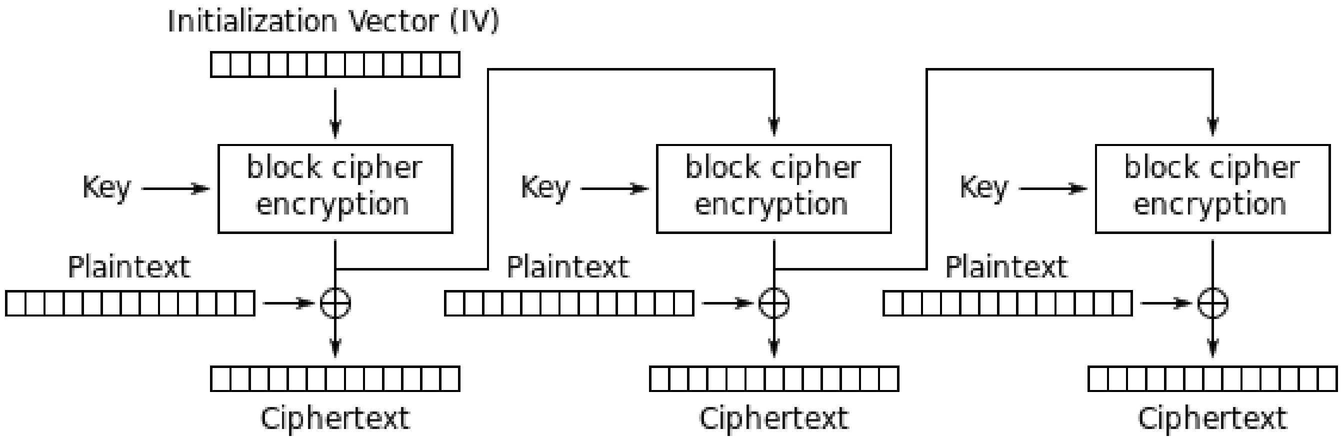
Output FeedBack (OFB)

- 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
- can be computed in advance

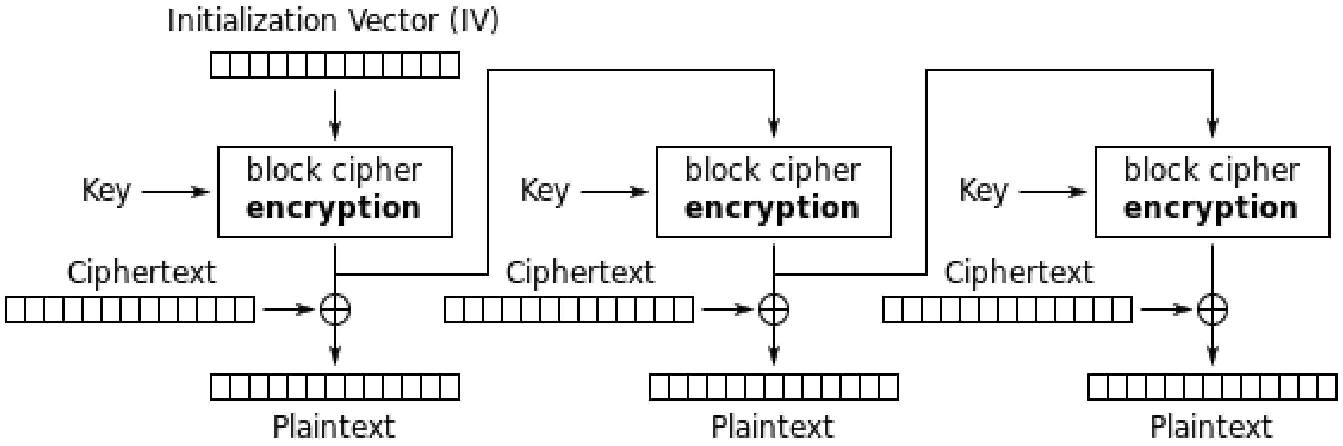
$$C_i = P_i \text{ XOR } O_i$$
$$O_i = DES_{K1}(O_{i-1})$$
$$O_{-1} = IV$$

- **uses:** stream encryption over noisy channels

Output FeedBack (OFB)

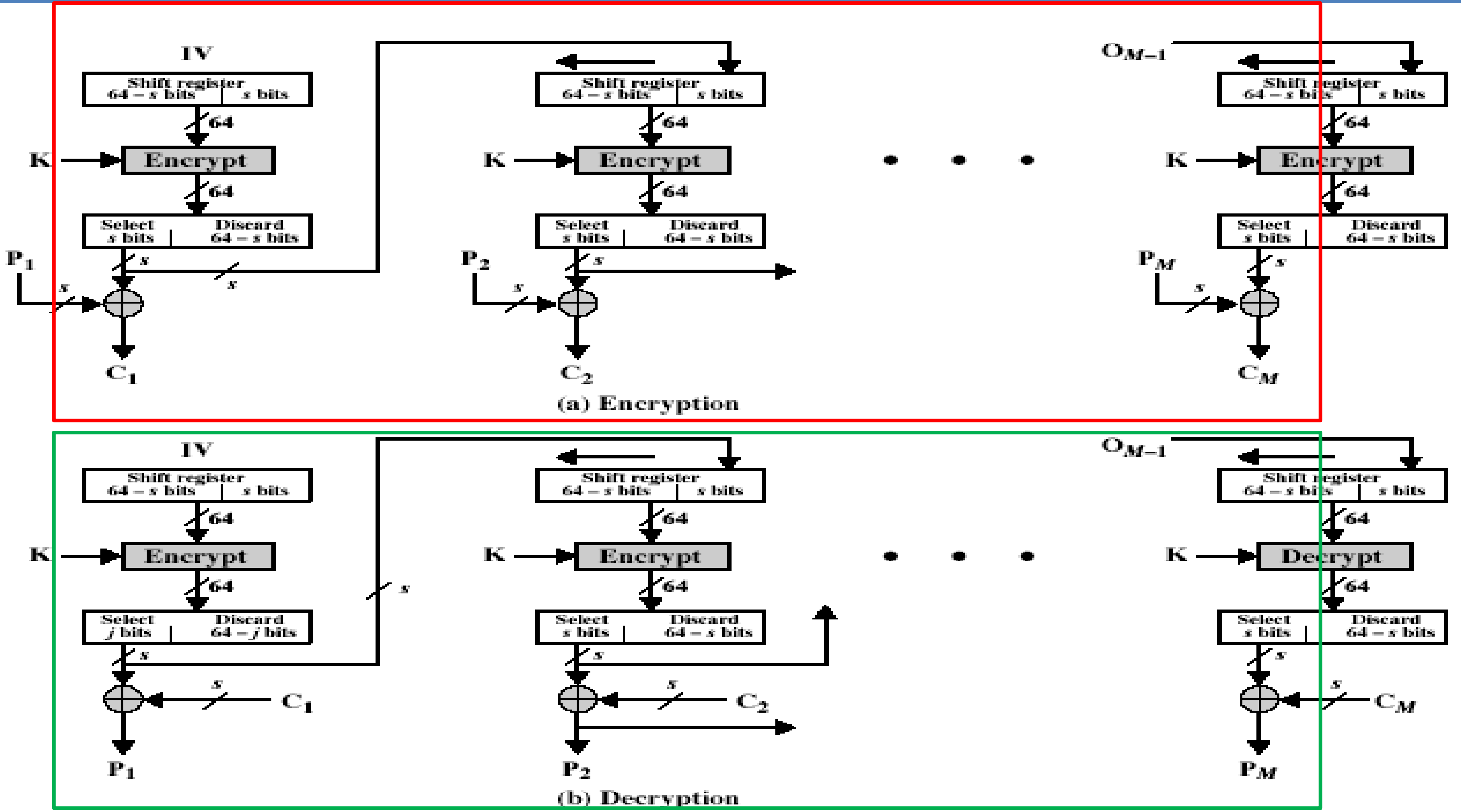


Output Feedback (OFB) mode encryption



Output Feedback (OFB) mode decryption

Output FeedBack (OFB)



Advantages and Limitations of OFB

- used when error feedback a problem or where need to encryptions before message is available
- superficially similar to CFB
- but feedback is from the output of cipher and is independent of message
- a variation of a **Vernam cipher**
  - hence must **never reuse the same sequence (key+IV)**
- sender and receiver must remain in sync, and some recovery method is needed to ensure this occurs
- originally specified with m-bit feedback in the standards
- subsequent research has shown that only **OFB-64** should ever be used

Advantages and Limitations of OFB

OFB	
Output Feedback	
Encryption parallelizable:	No
Decryption parallelizable:	No
Random read access:	No

Counter (CTR)

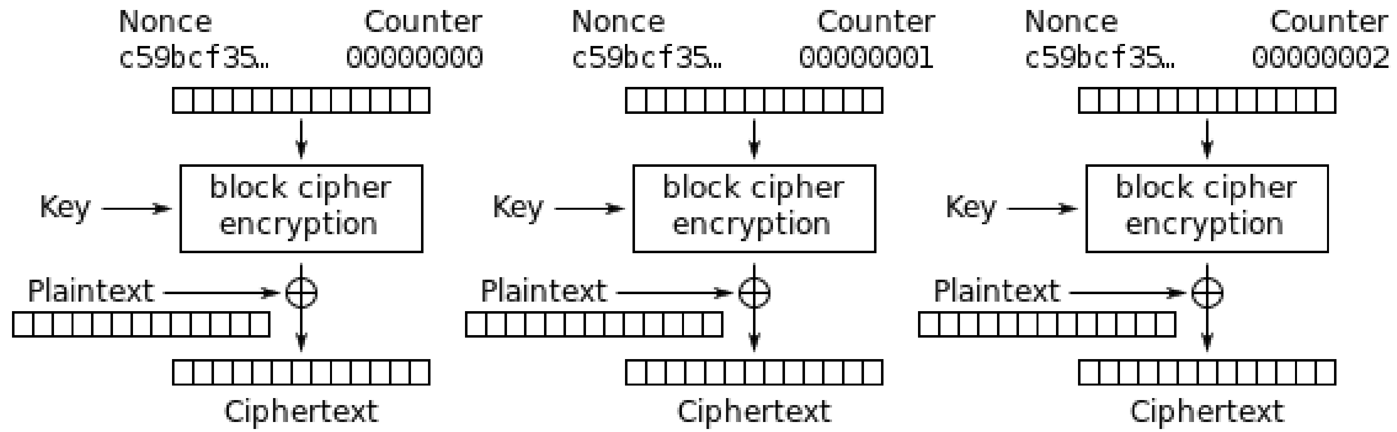
- a “new” mode, though proposed early on
- similar to OFB but encrypts counter value rather than any feedback value
- must have a **different key & counter value for every plaintext block** (never reused)

$$C_i = P_i \text{ XOR } O_i$$

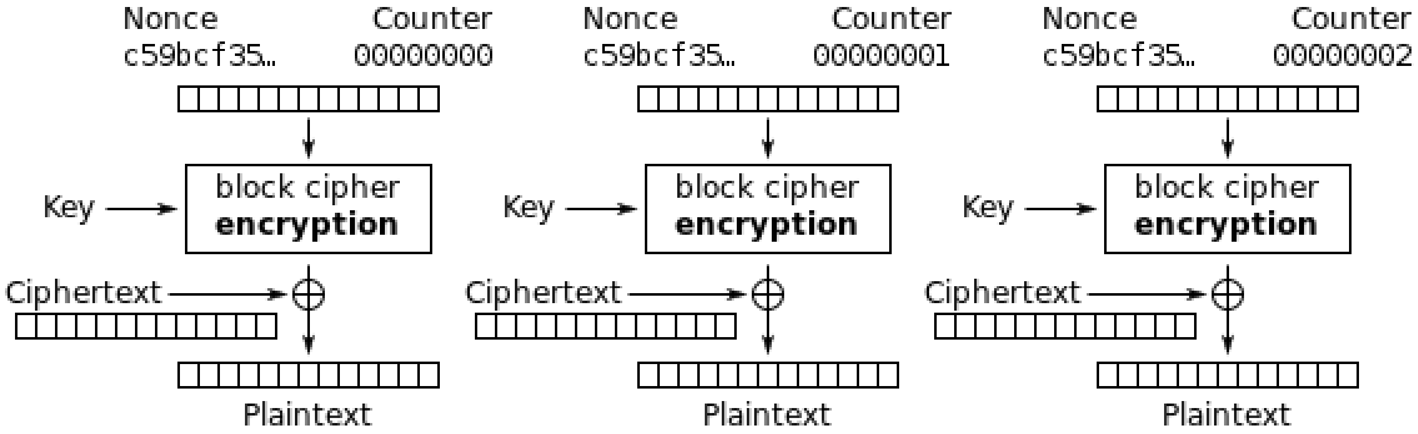
$$O_i = DES_{K1}(i)$$

- uses: high-speed network encryptions

Counter (CTR)

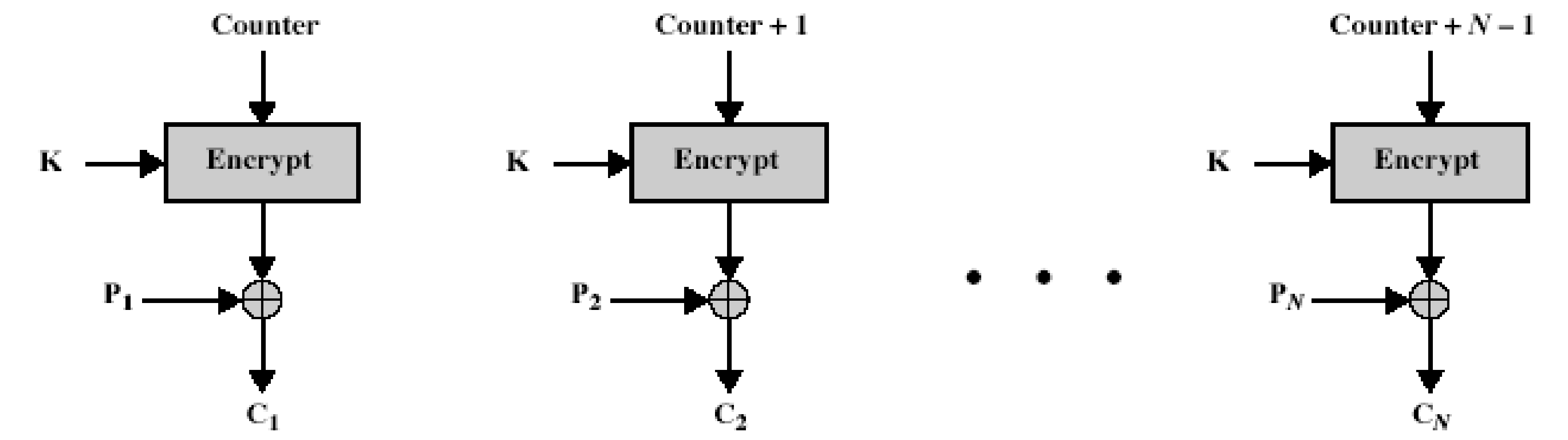


Counter (CTR) mode encryption

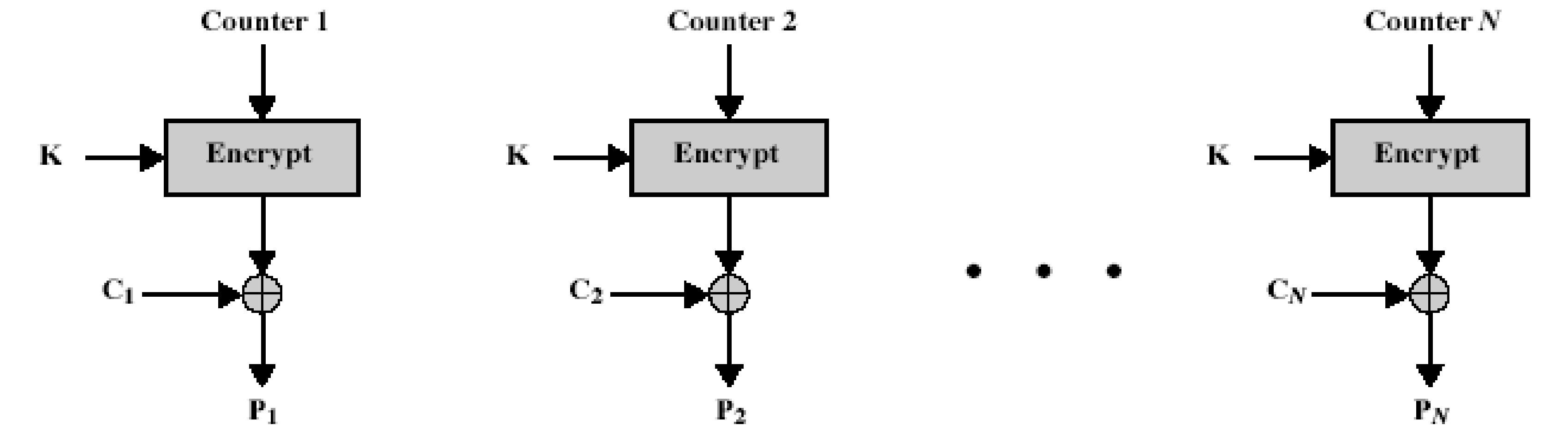


Counter (CTR) mode decryption

Counter (CTR)



(a) Encryption



(b) Decryption

Advantages and Limitations of CTR

- efficiency
  - can do parallel encryptions
  - 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)

Advantages and Limitations of CTR

CTR	
Counter	
Encryption parallelizable:	Yes
Decryption parallelizable:	Yes
Random read access:	Yes

Multiple DES

The major criticism of DES regards its key length. Fortunately DES is not a group. This means that we can use double or triple DES to increase the key size.

Topics discussed in this section:

Double DES
Triple DES

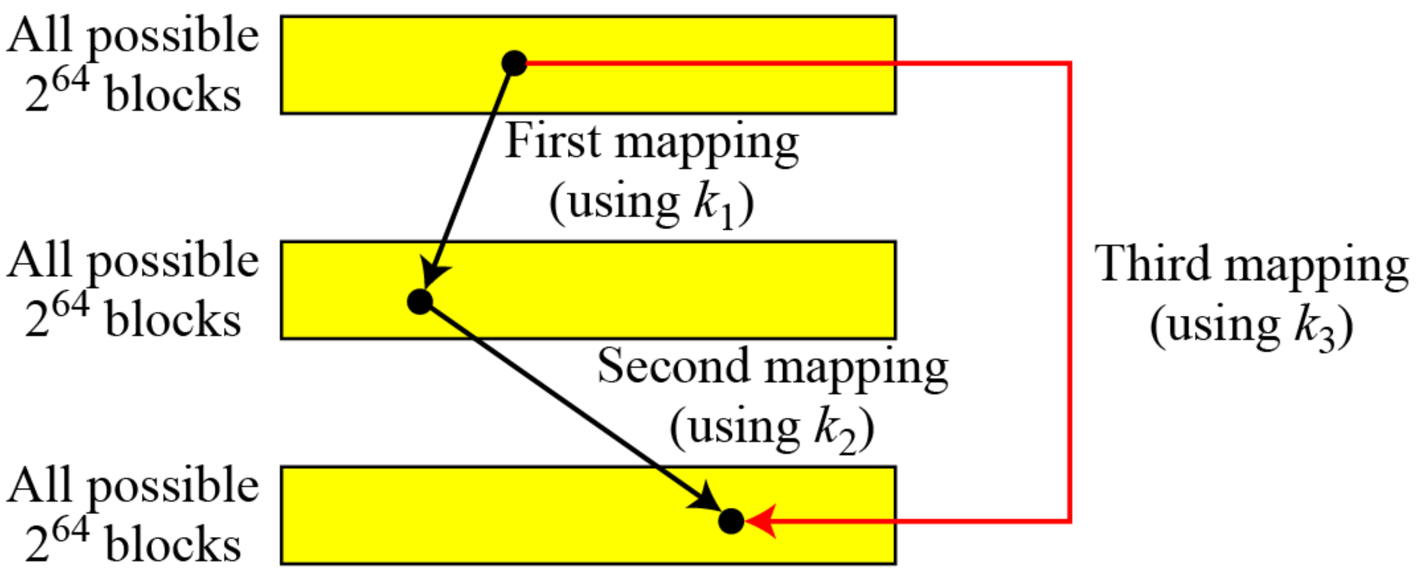
Multiple Encryption & DES

- 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

A substitution that maps every possible input to every possible output is a group.

Composition of mapping



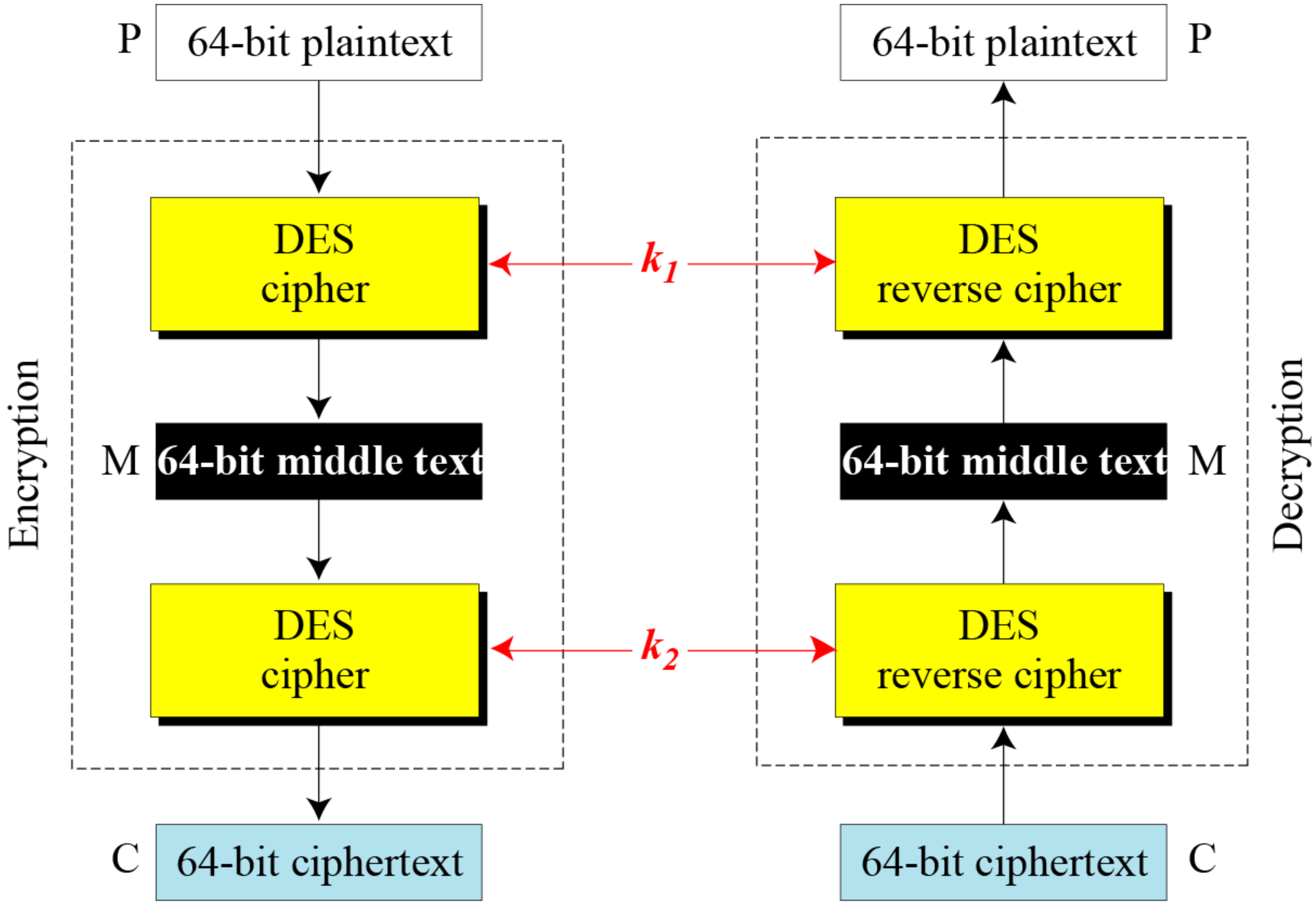
Double-DES?

- could use 2 DES encrypts on each block
  - $C = E_{K2}(E_{K1}(P))$
- issue of reduction to single stage
- **"meet-in-the-middle"** attack (Diffie in 1977)
  - works whenever use a cipher twice
  - since $X = E_{K1}(P) = D_{K2}(C)$
  - attack by encrypting P with all keys and store
  - then decrypt C with keys and match X value
  - can show takes $O(2^{56})$ **steps**

Meet-in-the-Middle Attack

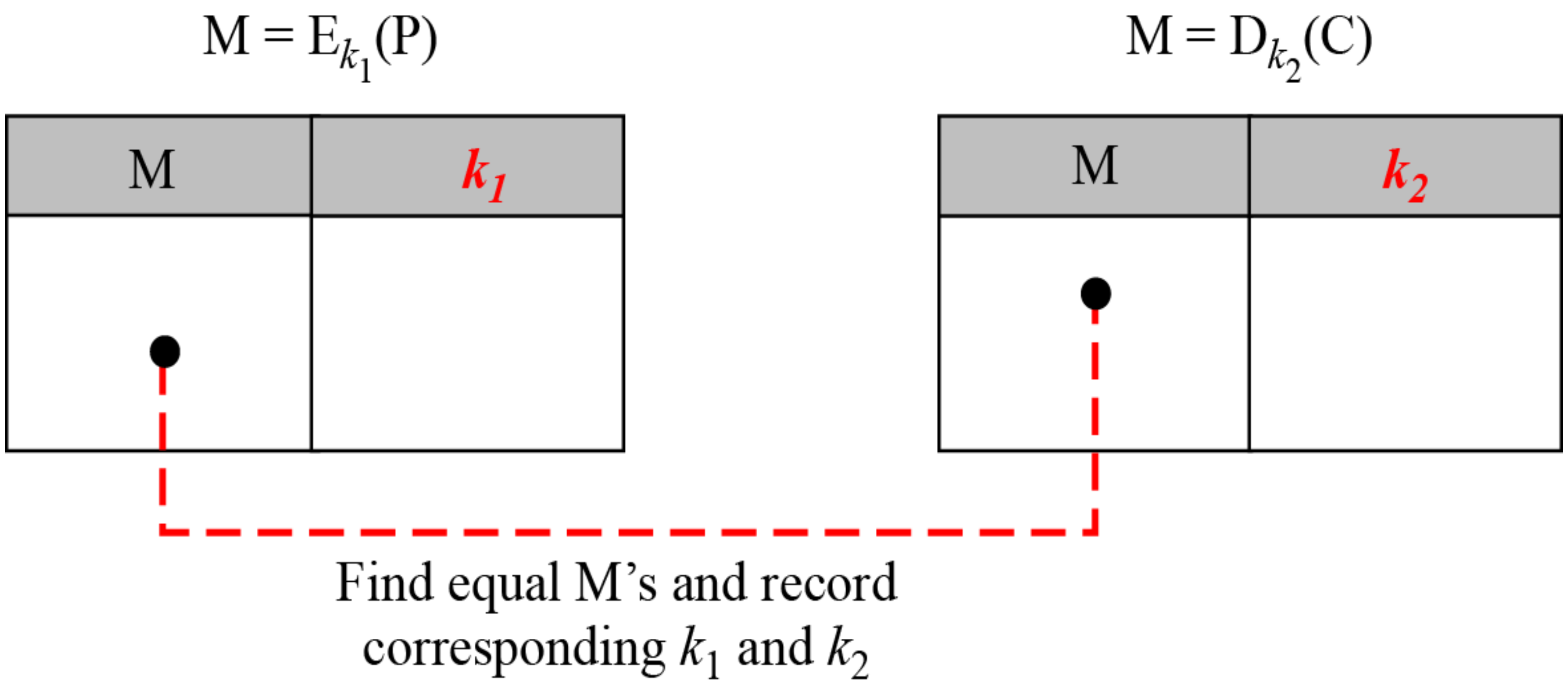
However, using a known-plaintext attack called meet-in-the-middle attack proves that double DES improves this vulnerability slightly (to 2^{57} tests), but not tremendously (to 2^{112}).

Meet-in-the-middle attack for double DES



Meet-in-the-Middle Attack

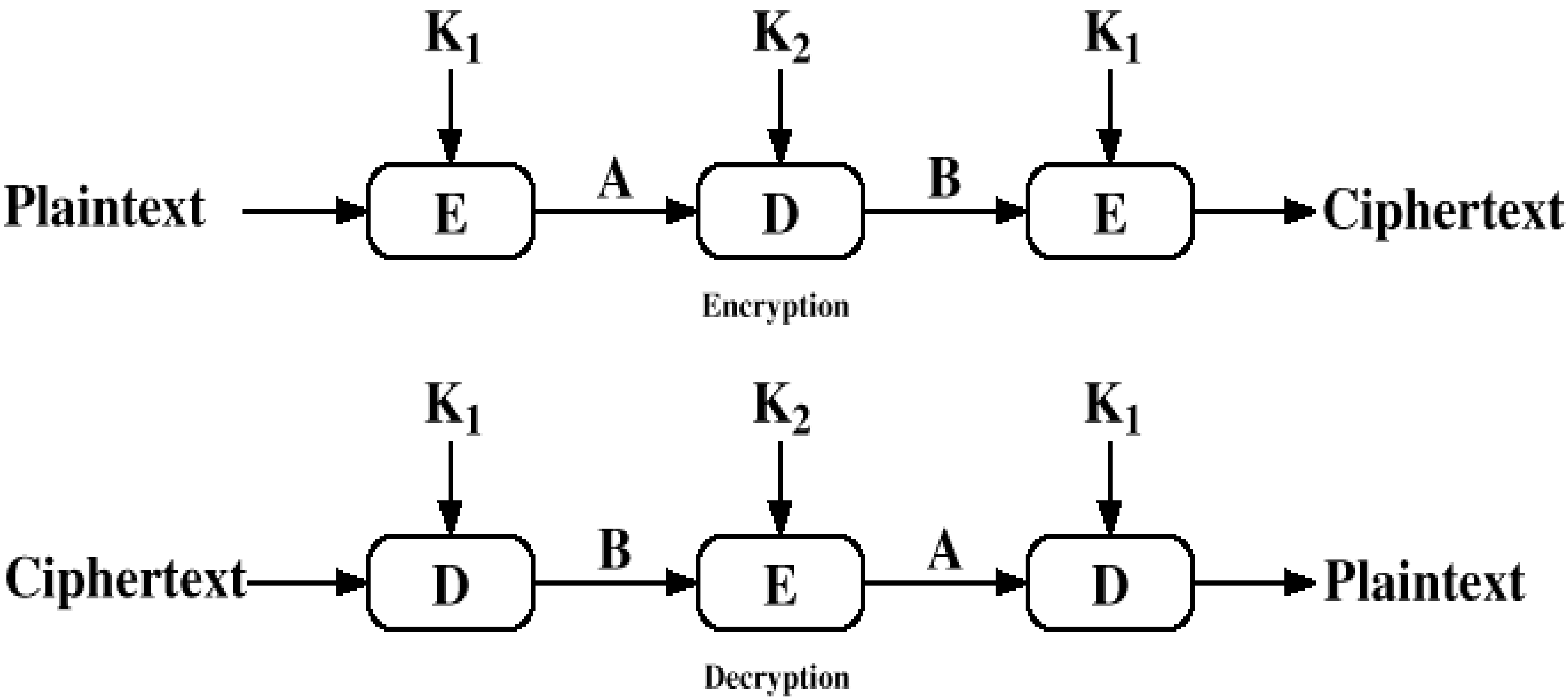
Tables for meet-in-the-middle attack



Triple-DES with Two-Keys

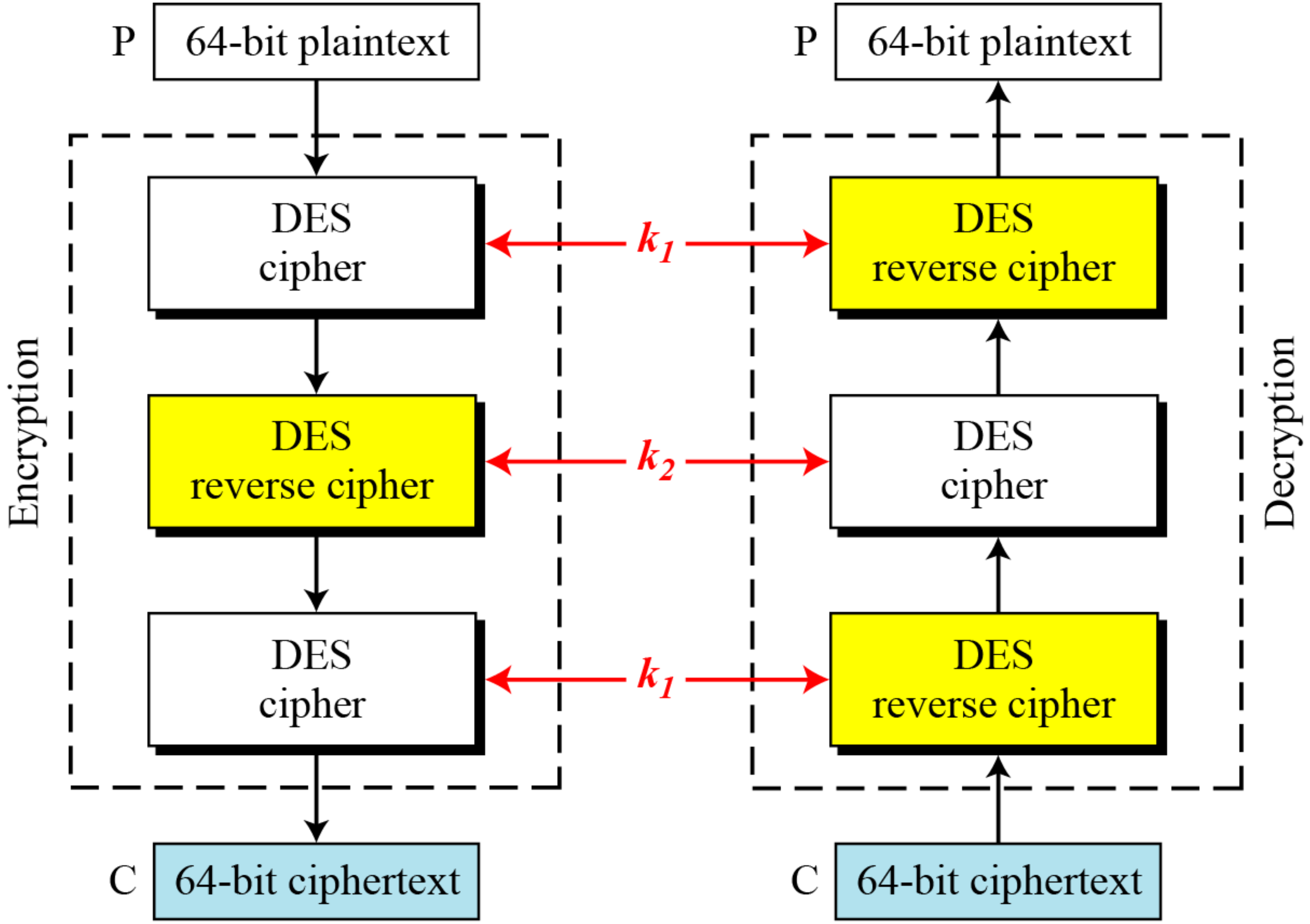
- hence must **use 3 encryptions**
  - would seem to need 3 distinct keys
- but can use 2 keys with E-D-E sequence
  - $C = E_{K1}(D_{K2}(E_{K1}(P)))$
  - nb encrypt & decrypt equivalent in security
  - **if K1=K2 then can work with single DES**
- standardized in ANSI X9.17 & ISO8732
- no current known practical attacks
  - several proposed impractical attacks might become basis of future attacks

Triple DES - More Secure



Triple DES

Figure 6.16 *Triple DES with two keys*



Triple DES with Three Keys

- The possibility of known-plaintext attacks on triple DES with two keys has enticed some applications to use triple DES with three keys.
- Triple DES with three keys is used by many applications such as PGP.

Summary

- have considered:
- block cipher design principles
- DES
  - details
  - strength
- Differential & Linear Cryptanalysis
- Modes of Operation
  - ECB, CBC, CFB, OFB, CTR

Dr. Lokesh Chouhan
NFSU Goa
Lokesh.chouhan_goa@nfsu.ac.in