CS 413
Information Security

Course Instructor

Naushad Siddiqui

Head of IT Audit, Sui Southern Gas Company

Visiting Faculty, NED University

Director & Faculty Member, ISACA Karachi

CS413 Information Security

Fall 2020

Week 08

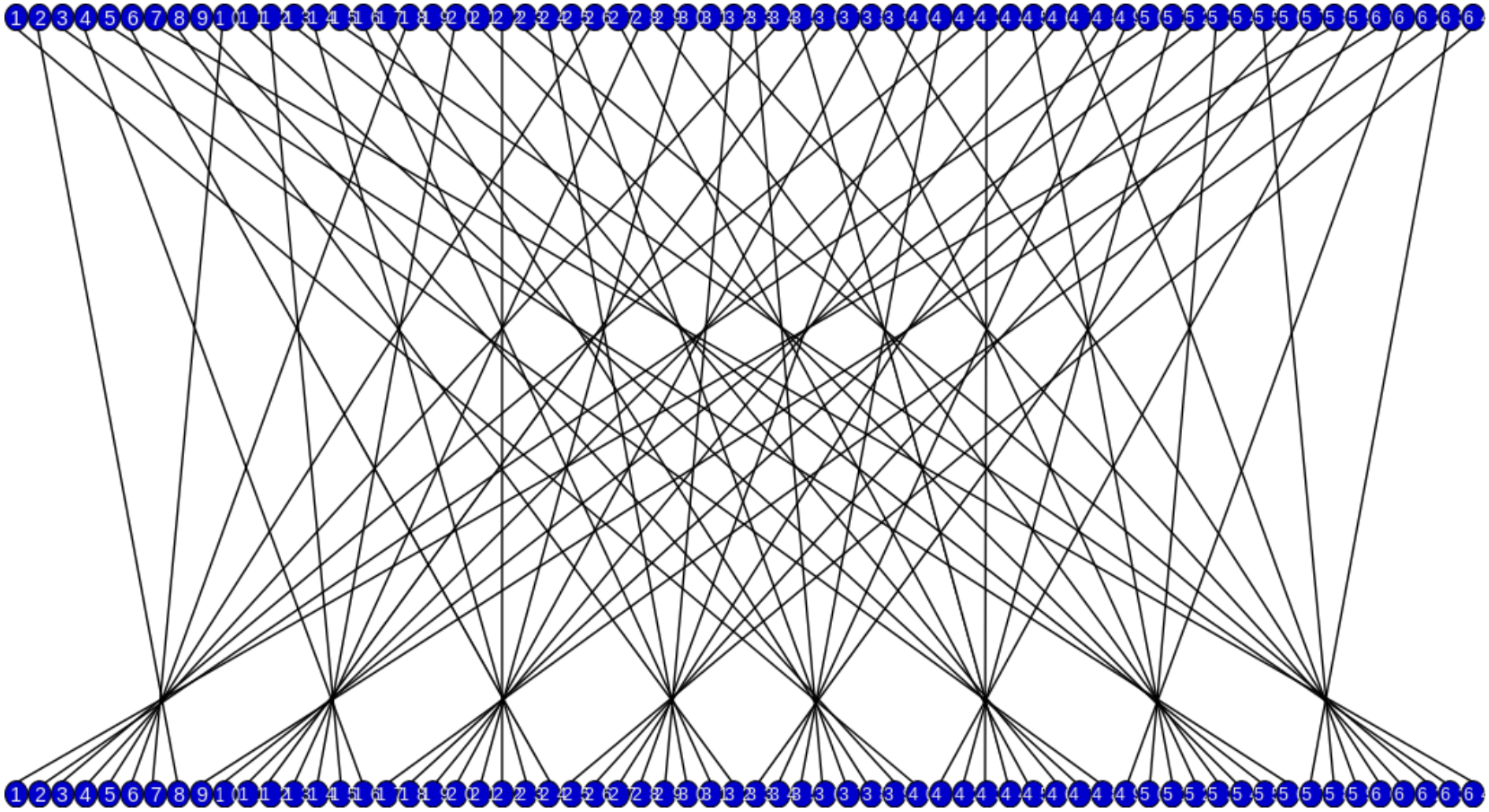
Agenda

- Round Keys Generation
- Initial Permutation (IP)
- Permuted Choices
- The Expansion Permutation (E)
- S-Box
- Avalanche Effect
- Multiple Encryption with DES

Round Keys Generation

- The 56 bit key size comes from security considerations
- It was big enough so that an exhaustive key search was about as hard as the best direct attack (a form of differential cryptanalysis called a T-attack, known by the IBM & NSA researchers), but no bigger.
- Main key is 64 bits. The extra 8 bits were then used as parity (error detecting) bits, which makes sense given the original design use for hardware communications links.
- 56-bits are selected and permuted using Permuted Choice One (PC1); and then divided into two 28-bit halves.
- In each round:
  - Left-rotate each half separately by either 1 or 2 bits according to a rotation schedule.
  - Select 24-bits from each half, and permute the combined 48 bits.
  - This forms a round key.

Initial Permutation (IP)



Initial Permutation (IP)

- IP: the first step of the encryption.
- It reorders the input data bits.
- The last step of encryption is the inverse of IP.
- IP and IP^{-1} are specified by tables

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

This table specifies the input permutation on a 64-bit block.
First bit of the output is taken from the 58th bit of the input; the second bit from the 50th bit, and so on, with the last bit of the output taken from the 7th bit of the input.

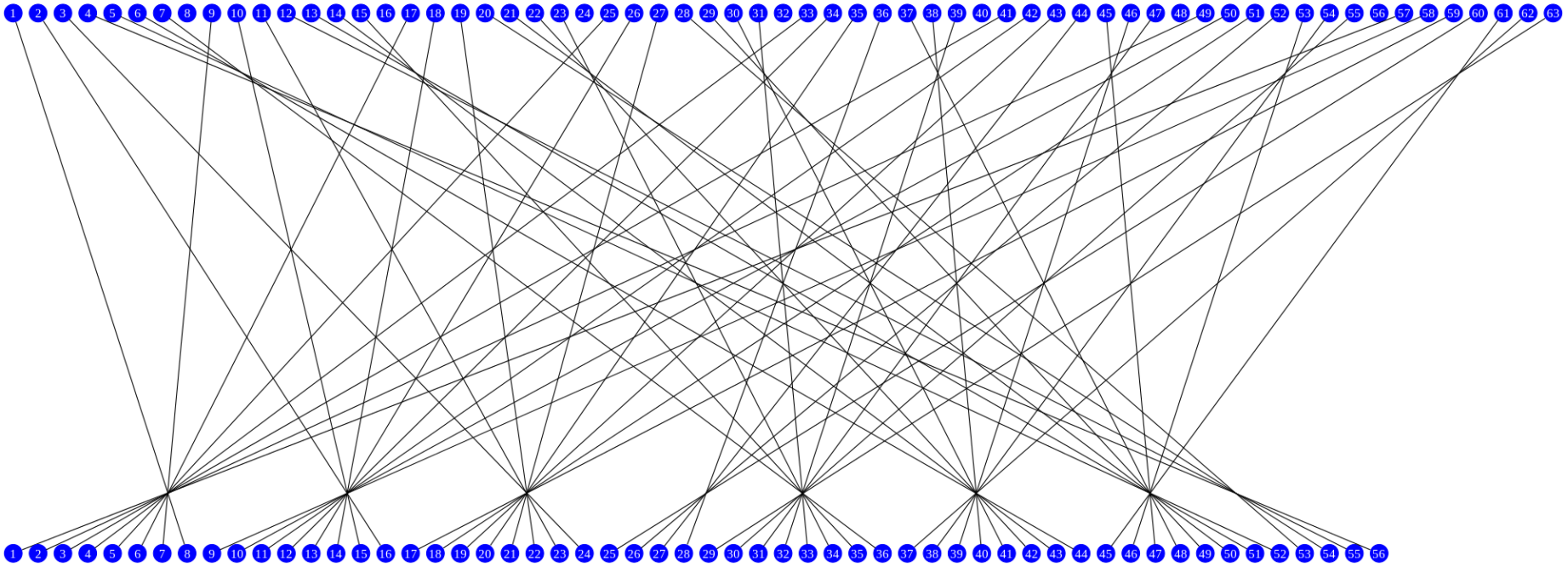
Final Permutation (IP^{-1})

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

The final permutation is the inverse of the initial permutation.
The table is interpreted similarly

Permuted Choice One (PC1)



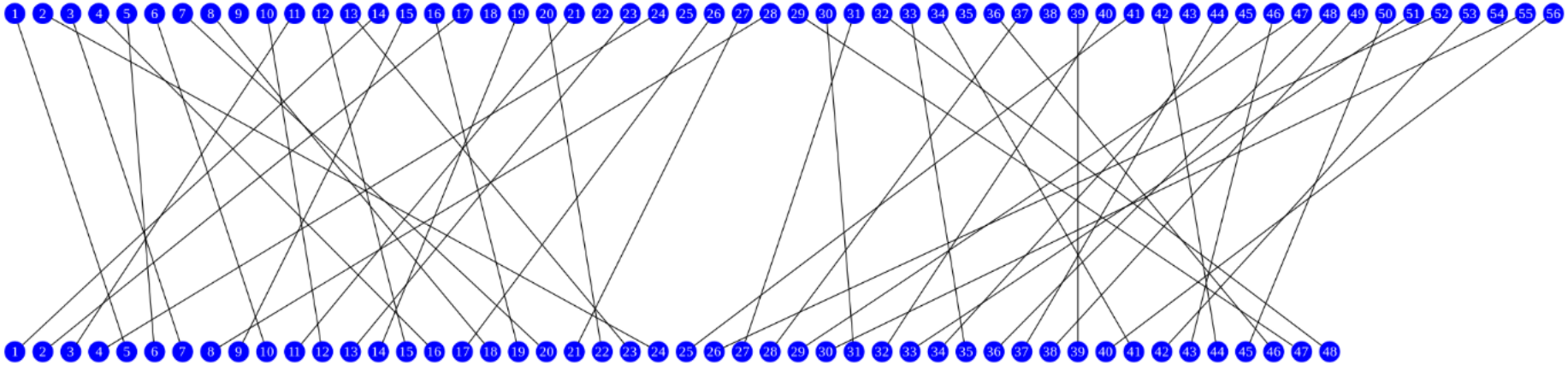
Permuted Choice One (PC1)

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

The "Left" and "Right" halves of the table show which bits from the input key form the left and right sections of the key schedule state.

Note that only 56 bits of the 64 bits of the input are selected; the remaining eight (8, 16, 24, 32, 40, 48, 56, 64) were specified for use as parity bits.

Permuted Choice Two (PC2)



This permutation selects the 48-bit subkey for each round from the 56-bit key-schedule state. This permutation will ignore 8 bits below:

Permuted Choice 2 "PC-2" Ignored bits 9,18,22,25,35,38,43,54.

Permuted Choice Two (PC2)

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

This permutation selects the 48-bit subkey for each round from the 56-bit key-schedule state. This permutation will ignore 8 bits below:

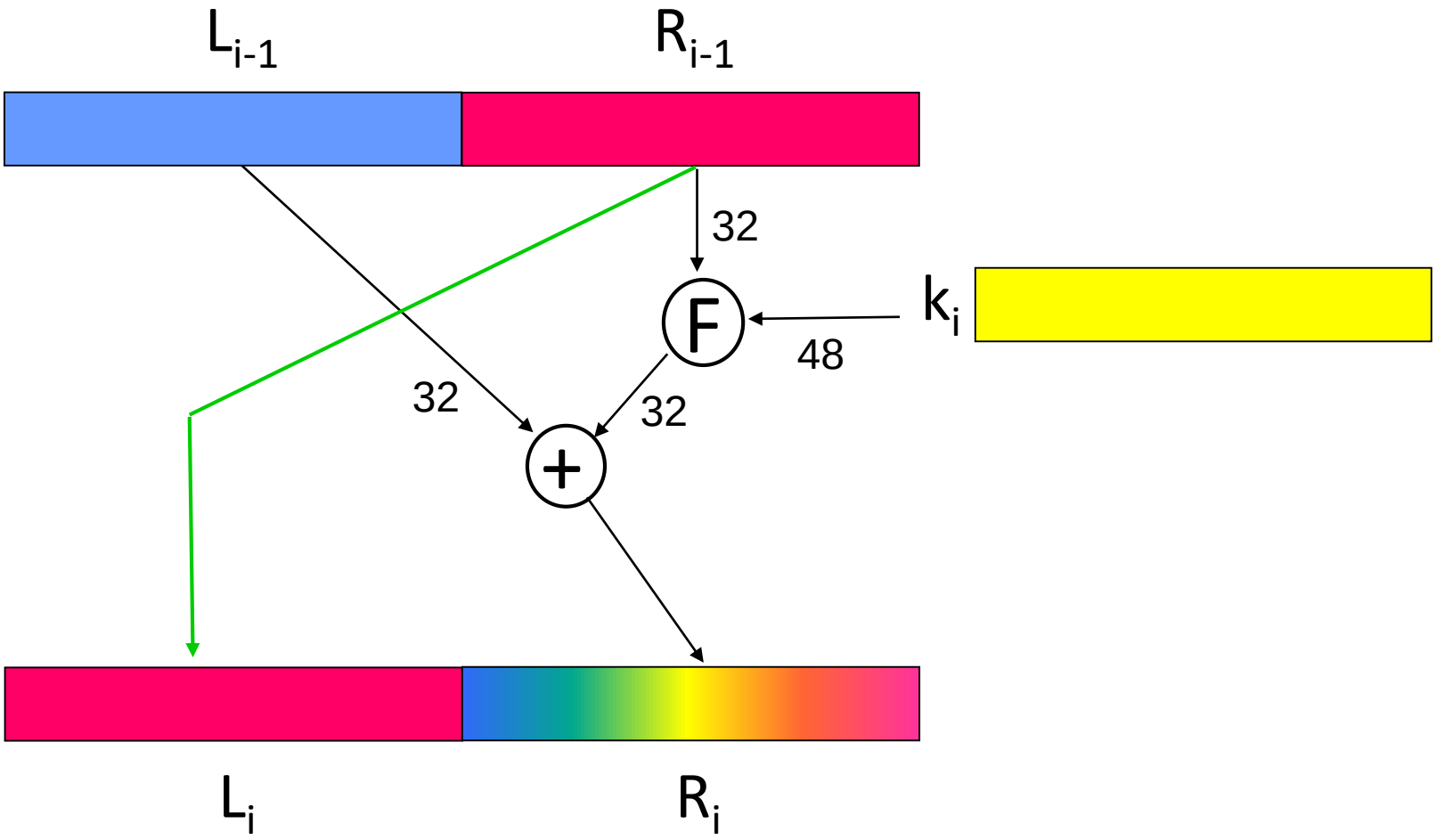
Permuted Choice 2 "PC-2" Ignored bits 9,18,22,25,35,38,43,54.

The 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

The Expansion function is interpreted as for the initial and final permutations. Some bits from the input are duplicated at the output; e.g. the fifth bit of the input is duplicated in both the sixth and eighth bit of the output. Thus, the 32-bit half-block is expanded to 48 bits.

Round i



The F Function of DES

The F function of DES

- The L and R each have 32 bits, and the round key K 48 bits.
- The F function, on input R and K, produces 32 bits:

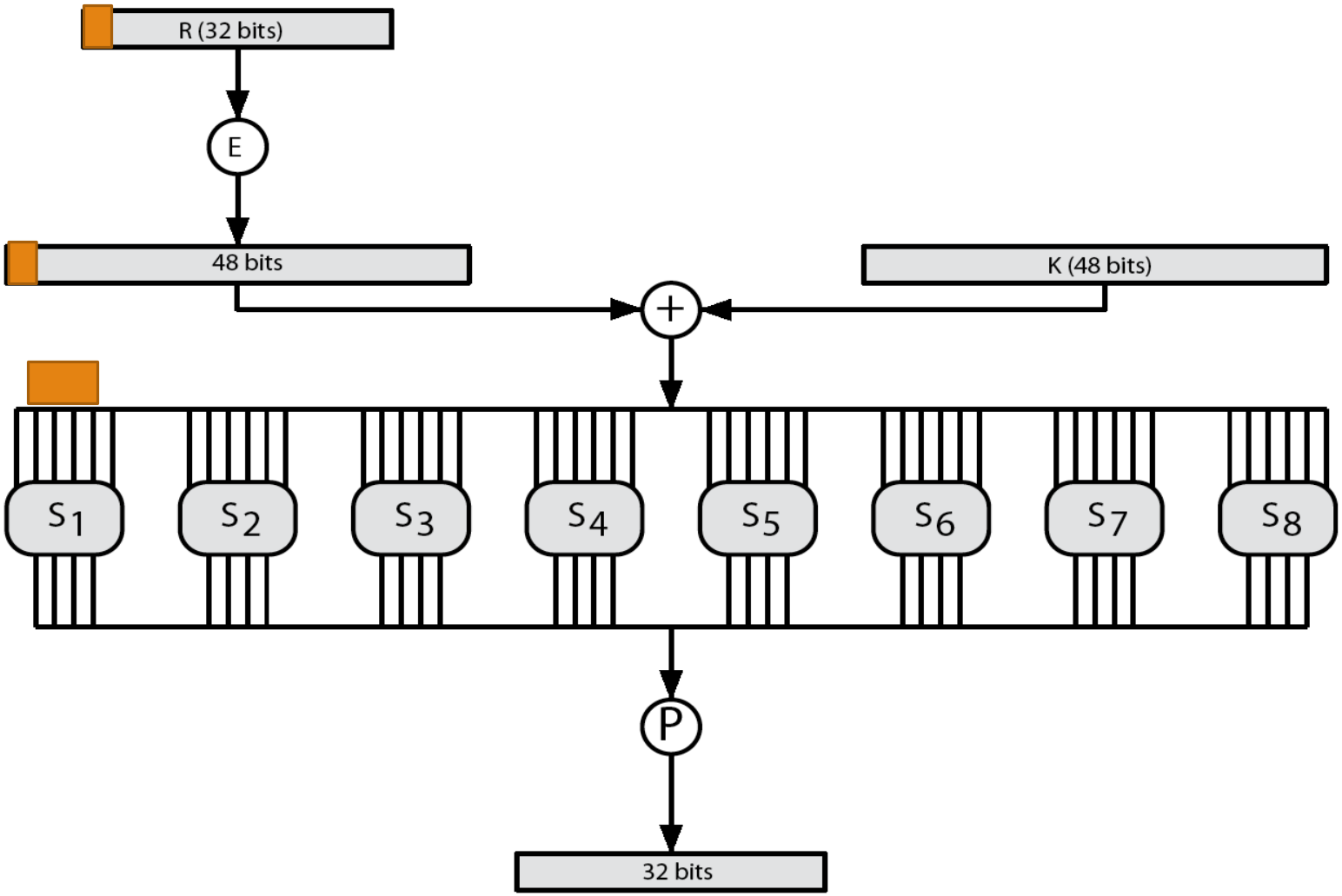
$$F(R, K) = P\big(S\big(E(R) \oplus K\big)\big)$$

where E: expands 32 bits to 48 bits;

S: shrinks it back to 32 bits;

P: permutes the 32 bits.

The F Function of DES



S-Box

- An S-box (substitution-box) is a basic component of symmetric key algorithms which performs substitution.
- In block ciphers, they are typically used to obscure the relationship between the key and the ciphertext - Shannon's property of confusion.
- In general, an S-box takes some number of input bits, ‘m’, and transforms them into some number of output bits, ‘n’, where ‘n’ is not necessarily equal to ‘m’.
- An m×n S-box can be implemented as a lookup table with 2m words of n bits each.
- Fixed tables are normally used, as in the Data Encryption Standard (DES), but in some ciphers the tables are generated dynamically from the key.

S-Box

- Eight S-boxes each map 6 to 4 bits
- Each S-box is specified as a 4 x 16 table
  - each row is a permutation of 0-15
  - outer bits 1 & 6 of input are used to select one of the four rows
  - inner 4 bits of input are used to select a column
- All the eight boxes are different.

One good example of a fixed table is the S-box from DES (S_5), mapping 6-bit input into a 4-bit output:

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

S-Box

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, $S_1(011011) = 9 = 1001$

Given a 6-bit input, the 4-bit output is found by selecting the row using the outer two bits (the first and last bits), and the column using the inner four bits. For example, an input "011011" has outer bits "01" and inner bits "1101"; the corresponding output would be "1001"

Avalanche Effect

- Avalanche effect:
  - A small change in the plaintext or in the key results in a significant change in the ciphertext.
  - an evidence of high degree of diffusion and confusion
  - a desirable property of any encryption algorithm

- DES exhibits a strong avalanche effect
  - Changing 1 bit in the plaintext affects 34 bits in the ciphertext on average.
  - 1-bit change in the key affects 35 bits in the ciphertext on average.

Attacks on DES

- Brute-force key search
  - Needs only two plaintext-ciphertext samples
  - Trying 1 key per microsecond would take 1000+ years on average, due to the large key space size, $256 \approx 7.2 \times 1016$.

- Differential cryptanalysis
  - Possible to find a key with 247 plaintext-ciphertext samples
  - Known-plaintext attack

- Liner cryptanalysis:
  - Possible to find a key with 243 plaintext-ciphertext samples
  - Known-plaintext attack

DES Cracker

- DES Cracker:
  - A DES key search machine
  - contains 1536 chips
  - Cost: $250,000.
  - could search 88 billion keys per second
  - won RSA Laboratory's "DES Challenge II-2" by successfully finding a DES key in 56 hours.
- DES is feeling its age. A more secure cipher is needed.

Multiple Encryption with DES

- In 2001, NIST published the Advanced Encryption Standard (AES) to replace DES.

- But users in commerce and finance are not ready to give up on DES.

- As a temporary solution to DES's security problem, one may encrypt a message (with DES) multiple times using multiple keys:
  - 2DES is not much securer than the regular DES
  - So, 3DES with either 2 or 3 keys is used

2DES

Consider 2DES with two keys:

$C = E_{K2}(E_{K1}(P))$

Decryption: $P = D_{K1}(D_{K2}(C))$

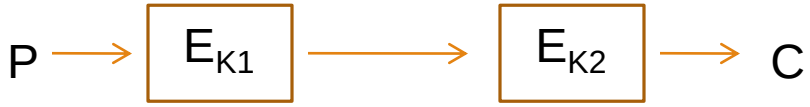
Key length: 56 x 2 = 112 bits

This should have thwarted brute-force attacks?

Wrong!

Meet-in-the-Middle Attack on 2DES

2-DES: $C = E_{K2}(E_{K1}(P))$



Given a known pair (P, C), attack as follows:

- Encrypt P with all 2^{56} possible keys for K1.
- Decrypt C with all 2^{56} possible keys for K2.
- If $E_{K1'}(P) = D_{K2'}(C)$, try the keys on another (P', C').
- If works, (K1', K2') = (K1, K2) with high probability.
- Takes $O(2^{56})$ steps; not much more than attacking 1-DES.

3DES with 2 Keys

3DES with 2 keys

- A straightforward implementation would be :

$$c := E_{k_1}\left(E_{k_2}\left(E_{k_1}(m)\right)\right)$$

- In practice : $c := E_{k_1}\left(D_{k_2}\left(E_{k_1}(m)\right)\right)$
  - Also referred to as EDE encryption
- Reason : if $k_1 = k_2$, then 3DES = 1DES.

  Thus, a 3DES software can be used as a single-DES.
- Standardized in ANSI X9.17 & ISO 8732.
- No practical attacks are known.

3DES with 3 Keys

3DES with 3 keys

- Encryption: $c := E_{k_3}\left(D_{k_2}\left(E_{k_1}(m)\right)\right)$.
- If $k_1 = k_3$, it becomes 3DES with 2 keys.
- If $k_1 = k_2 = k_3$, it becomes the regular DES.
- So, it is backward compatible with both 3DES with 2 keys and the regular DES.
- Some internet applications adopt 3DES with three keys; e.g. PGP and S / MIME.

End of Week 08