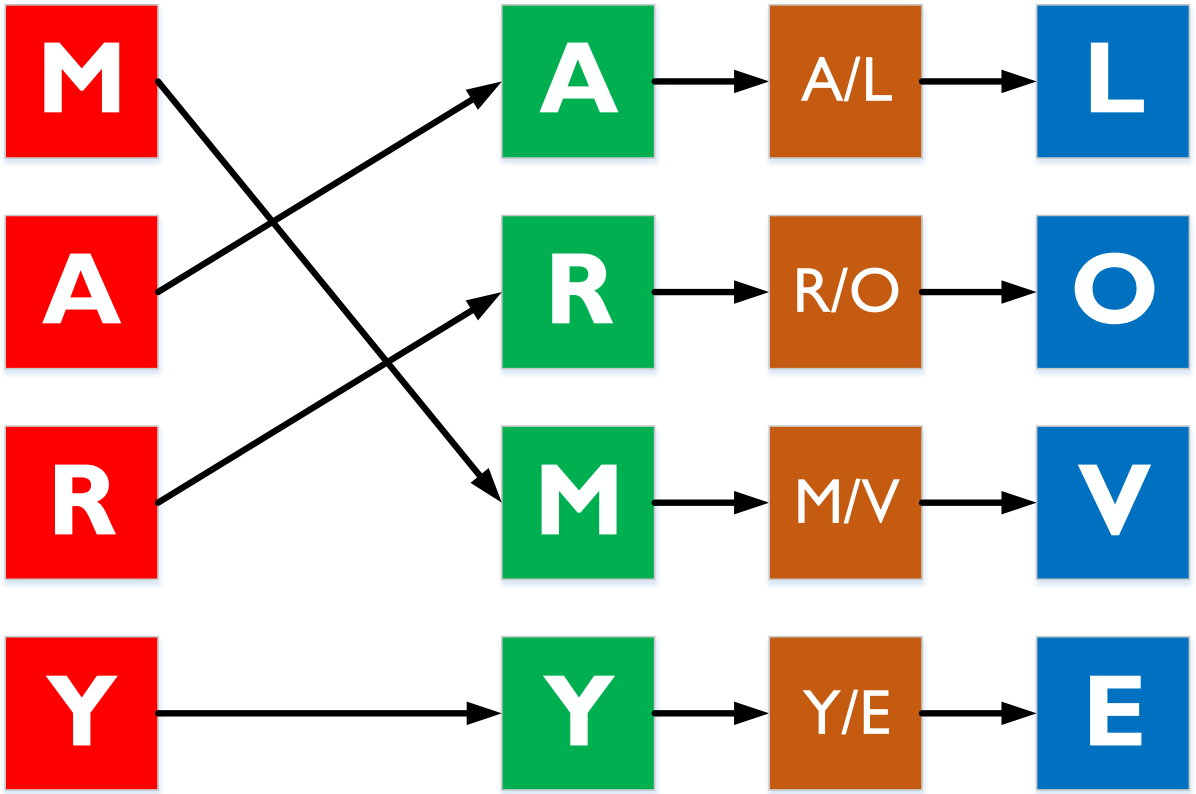
Last Time

- Cryptosystem (*E, D, M, K, C*)
- Secret Key Cryptography: a single key is used for both encryption and decryption algorithms
- Transposition Cipher & Substitution Cipher
- Brute-force attack
- One-Time Pad: random bit string as key, XOR, one-time

Exercise

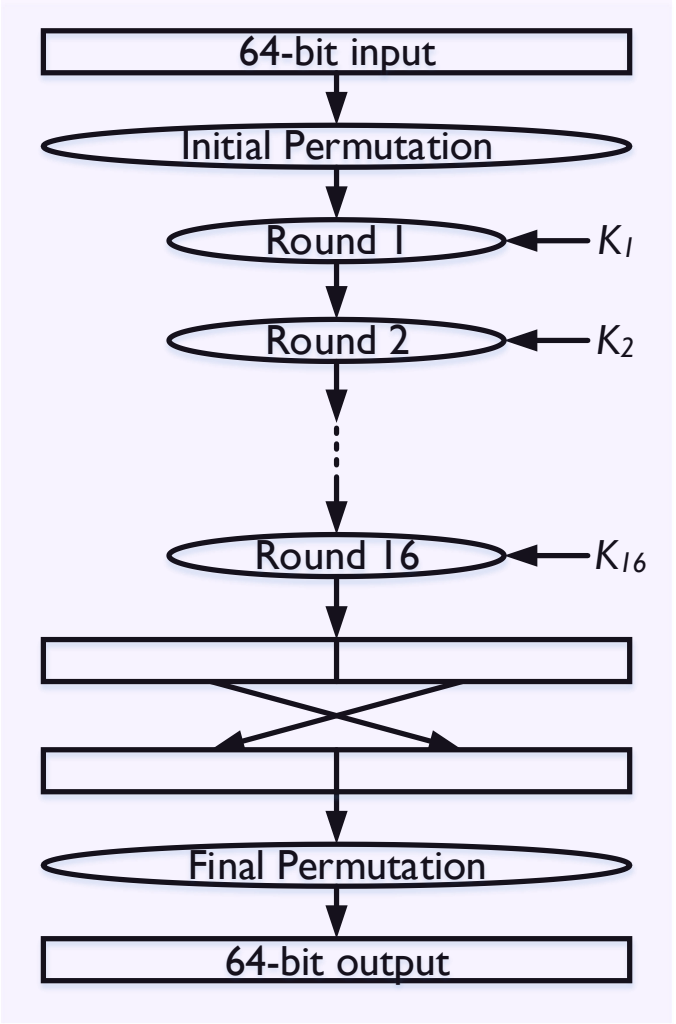
- Transposition cipher + Substitution cipher



- Q: Will the combination of two ciphers enhance security?

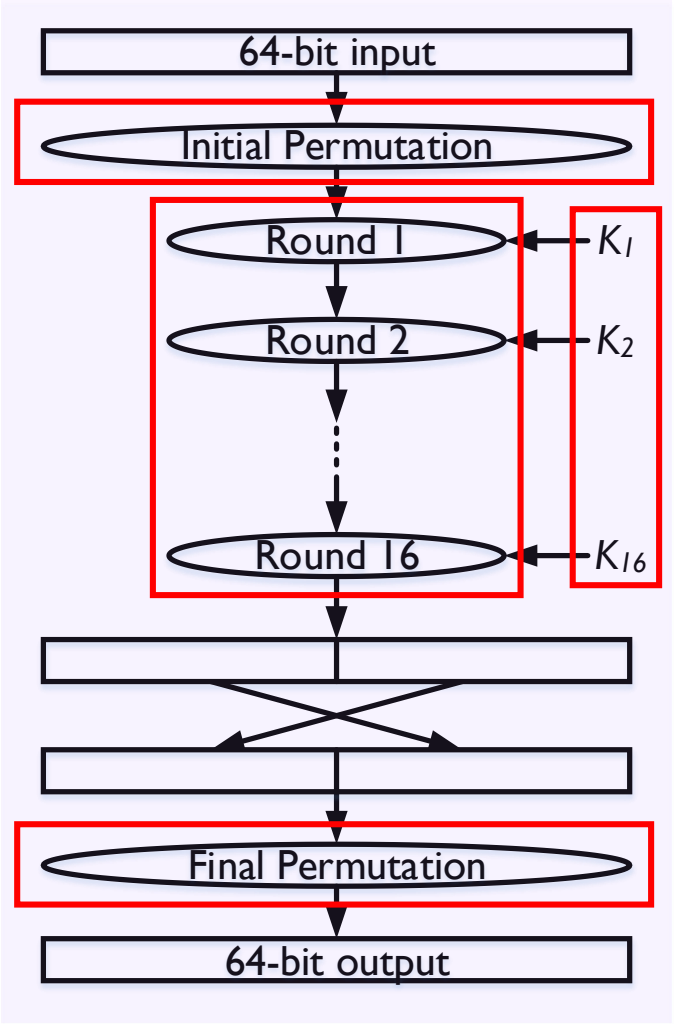
Data Encryption Standard

- Designed by IBM and published by the US NBS (now NIST) in 1977
- Signaled the beginning of the modern area of cryptography
- Block cipher: map a 64-bit input block to a 64-bit output block by using a 64-bit key (56 bits+8 parity bits)



DES Overview

- Initial and Final Permutations
  - Final permutation is the inverse of the initial permutation
- Per-round Key Generation
  - 16 48-bit per-round keys
- Per-round Encryption



Data Permutations

Initial Permutation

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

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

Data Permutations

Final Permutation

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

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

Per-Round Key Generation

- Initial Permutation
  - Bits 8, 16, 24, …, 64 are discarded

C_0

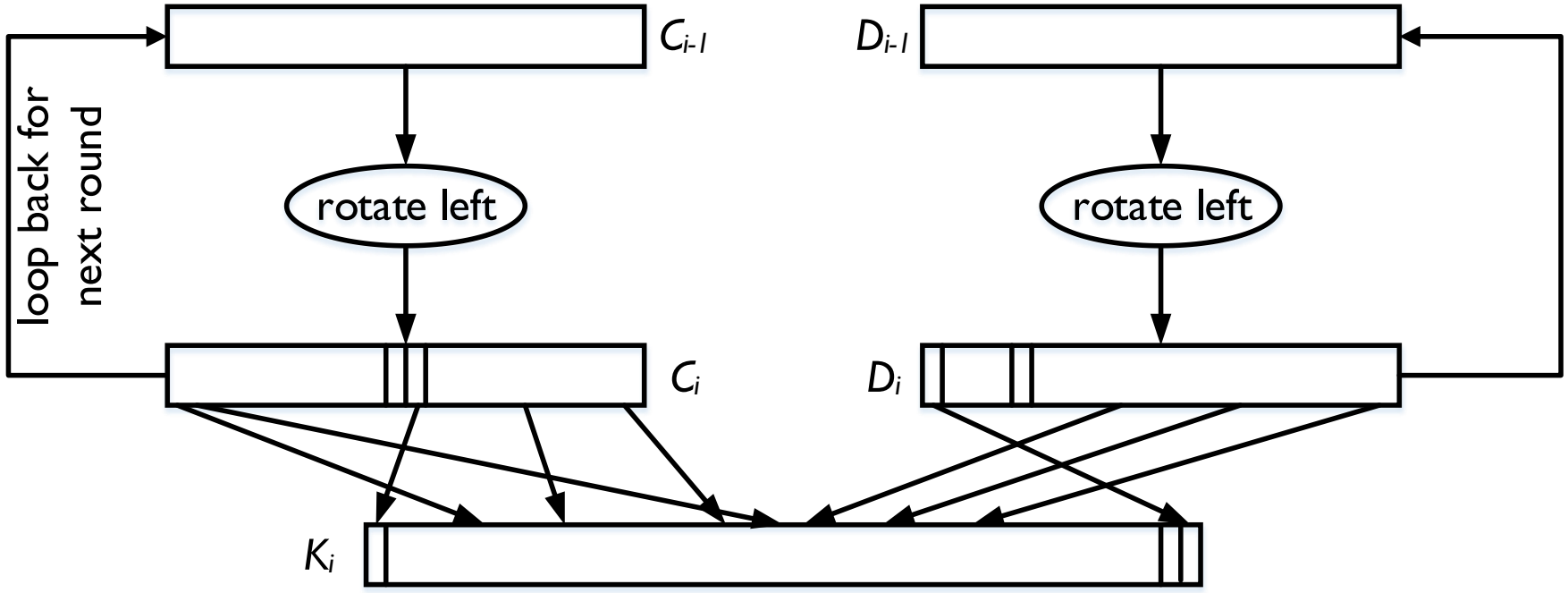
57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36

D_0

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

Per-Round Key Generation

- Round *i* for generating K_i
- Left rotation
  - In rounds 1, 2, 9, and 16, circular shift of 1 bit to the left
  - In other rounds, circular shift of 2 bit to the left



Per-Round Key Generation

- Per-round Permutation
  - Bits 9, 18, 22, 25, 35, 38, 43, and 54 are discarded
  - Per-round key K_i is 48-bit long

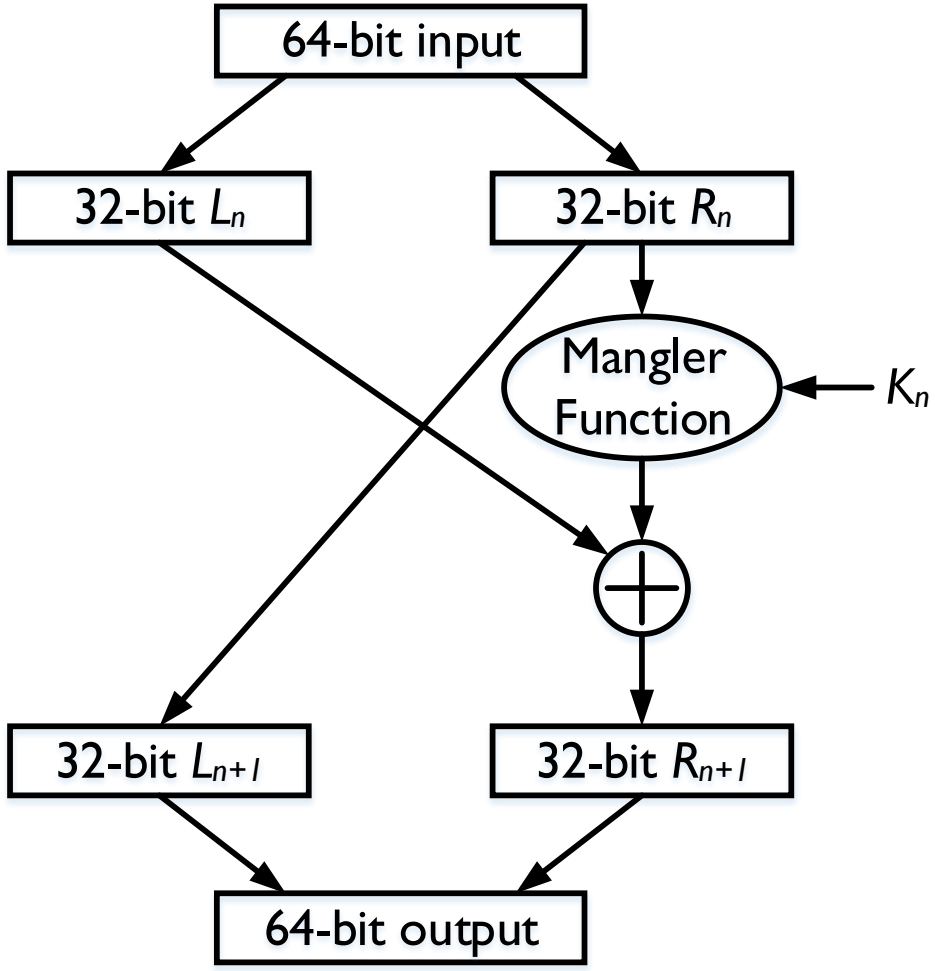
left half of K_i

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2

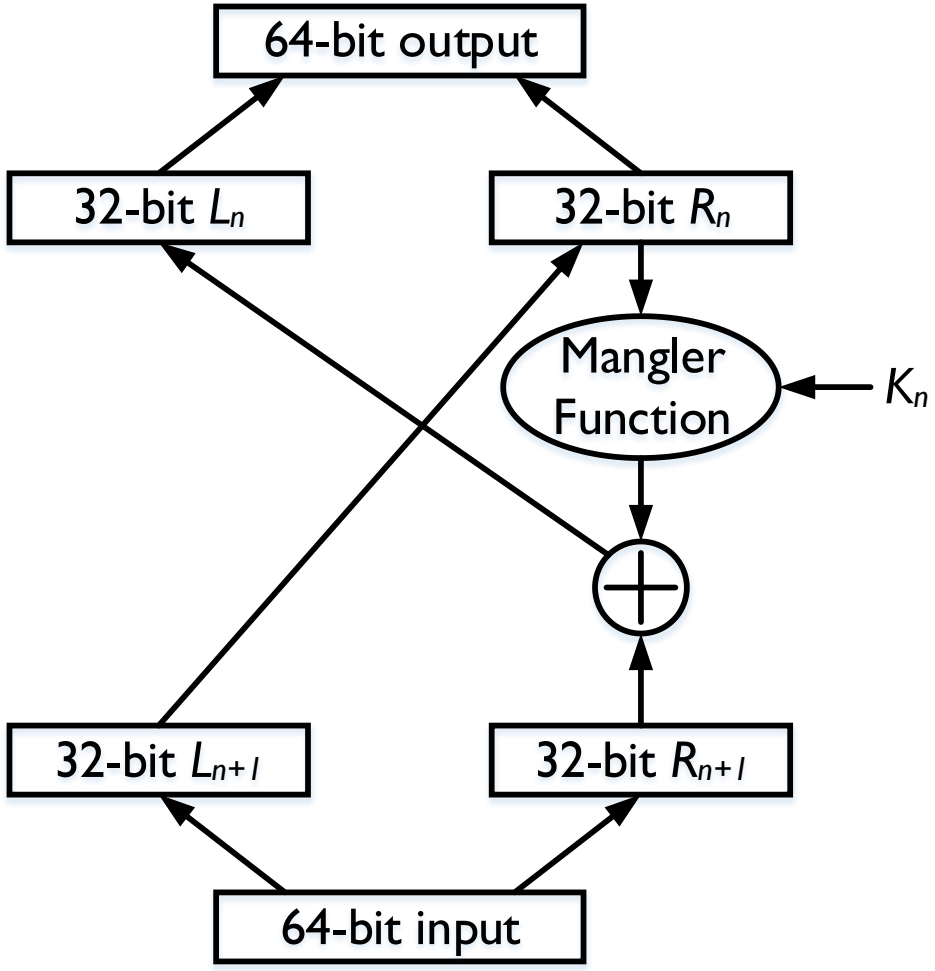
right half of K_i

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 Round



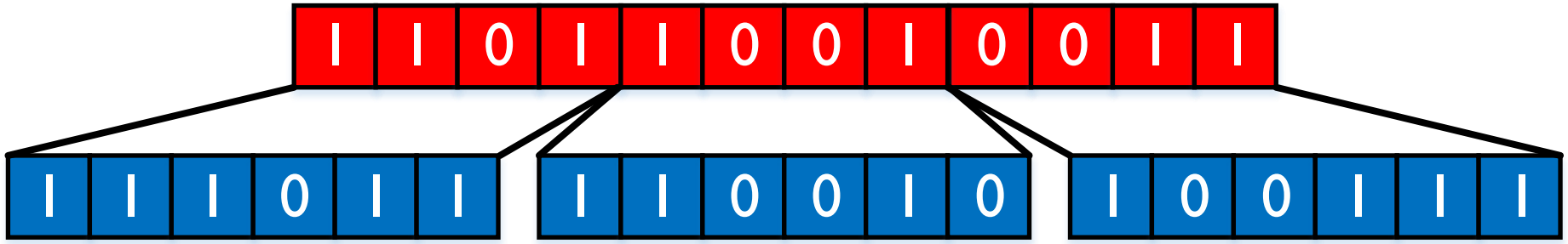
Encryption Algorithm



Decryption Algorithm

Mangler Function

- STEP 1: Expand R_n from a 32-bit value to a 48-bit value
  - Break R_n into 8 4-bit chunks
  - Expand each chunk to 6 bits by concatenating adjacent bits to it



- STEP 2: Break 48-bit K_i into 8 6-bit chunks
- STEP 3: XOR the 8 chunks of R_n and K_i, and feed the results into 8 substitution boxes to generate 32-bit output
- STEP 4: Final Permutation

Mangler Function

- A substitution box (or S-box) is used to obscure the relationship between the plaintext and the ciphertext
  - In DES S-boxes are carefully chosen to resist cryptanalysis

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

Example: Given a 6-bit input, the 4-bit output is found by selecting the row using the outer two 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".

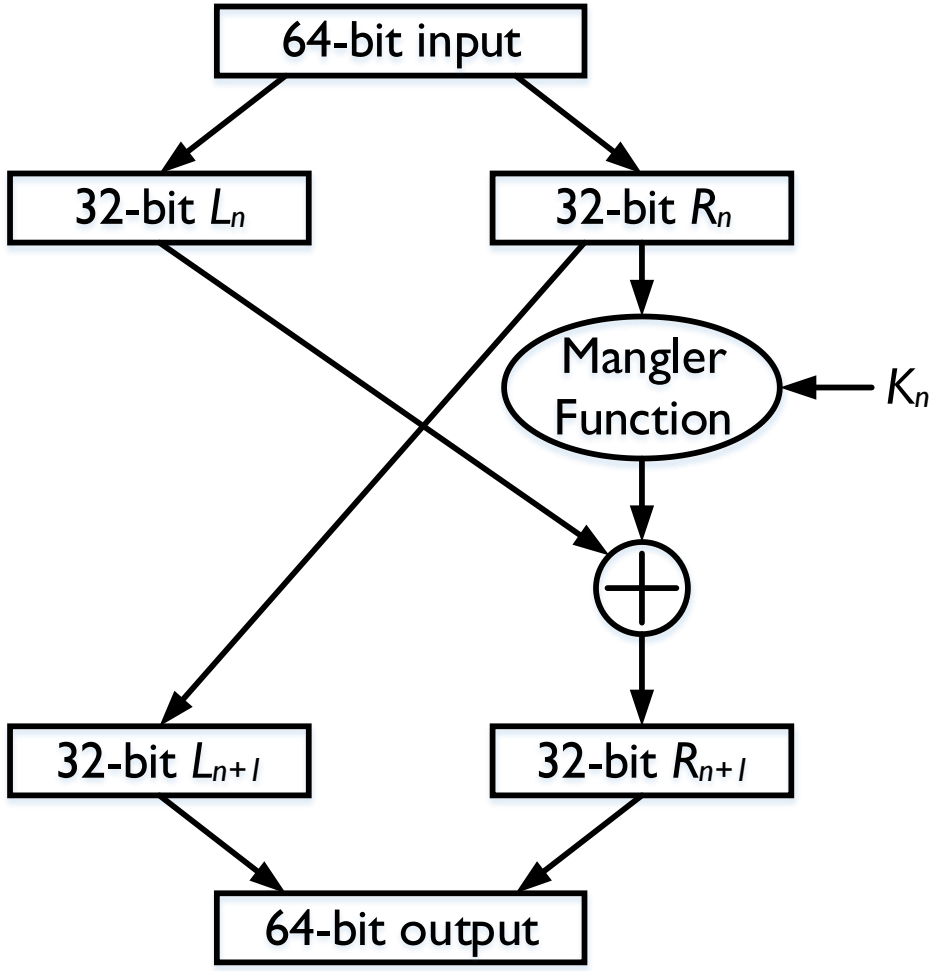
Mangler Function

- Final permutation on the bits in the 32-bit output of 8 substitution boxes

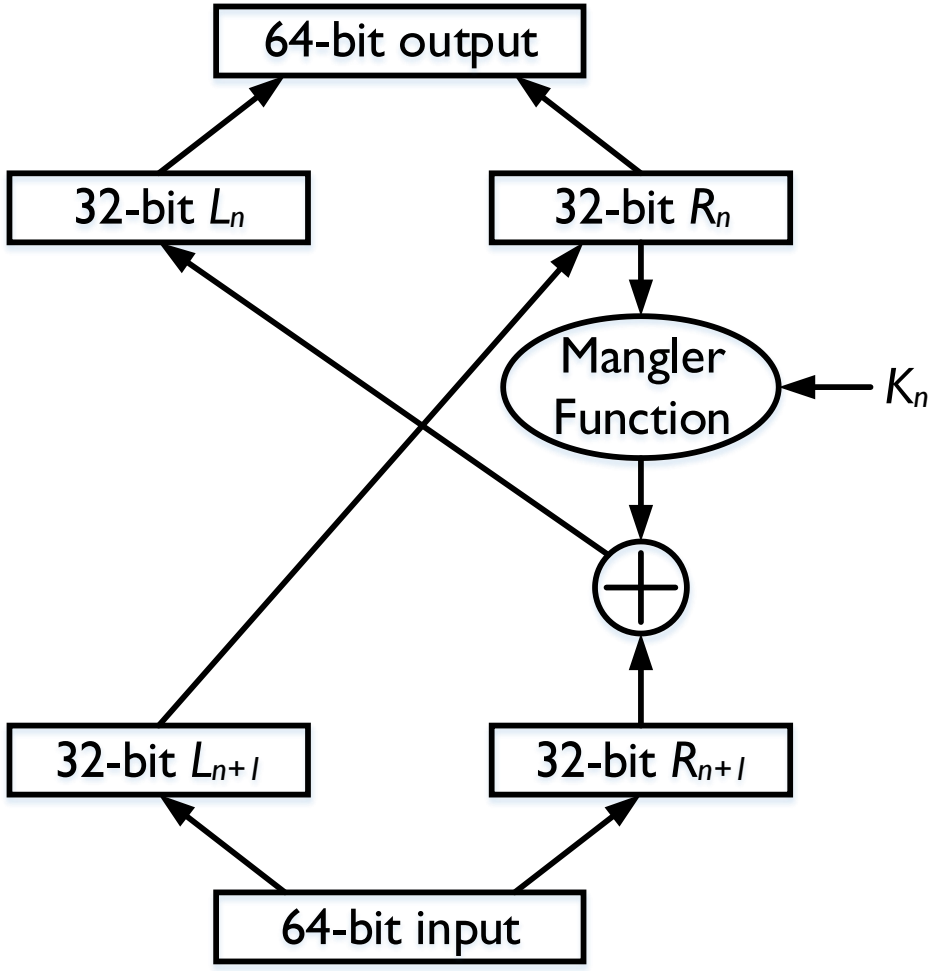
1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32

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

DES Round



Encryption Algorithm



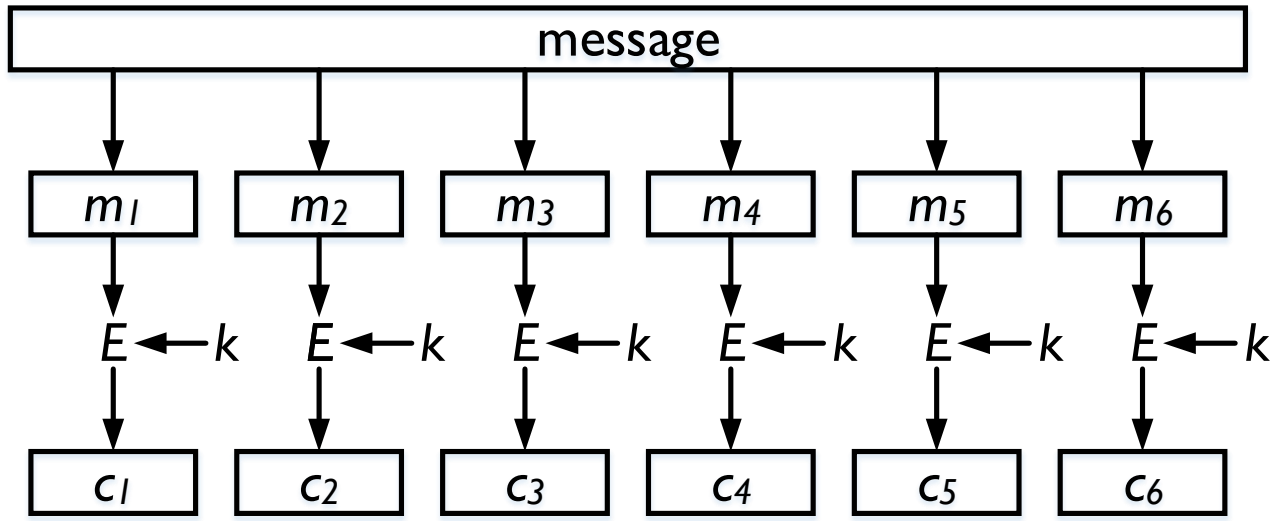
Decryption Algorithm

Cryptanalysis of DES

- DES has an effective 56-bit key length
  - Wiener: 1,000,000$ - 3.5 hours (never built)
  - July 17, 1998, the EFF DES Cracker, which was built for less than $250,000 < 3 days
  - January 19, 1999, Distributed.Net (w/EFF), 22 hours and 15 minutes (over nearly 100,000 machines)
  - September 2002, FPGA implementation, 12 hours
  - We all assume that NSA and agencies like it around the world can crack (recover key) DES in milliseconds
- What now? Give up on DES?

Electronic Code Block

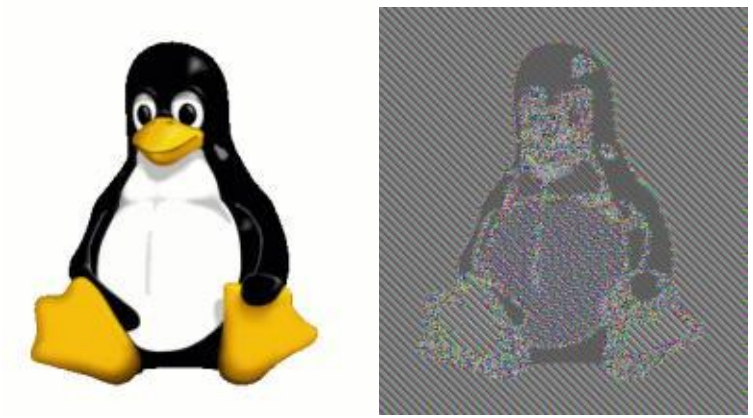
- How to encrypt a message with length > 64 bits?
- Electronic Code Block (ECB)
  - Message is broken into 64-bit blocks (padding the last one if needed)
  - Each block is independently encrypted using DES with the same key



ECB Encryption Algorithm

Pros and Cons

- Error in one received ciphertext block does not affect the correct decryption of other ciphertext blocks
- Identical plaintext blocks produce identical ciphertext blocks resulting in recognizable pattern

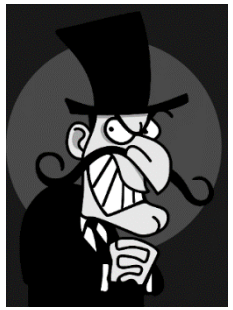


- Ciphertext blocks can be easily rearranged

Rearranging Attack

HR

First Name	Last Name	Position	Salary
Adams	Wong	President	300,000
Hao	Yue	Assistant Professor	50,000

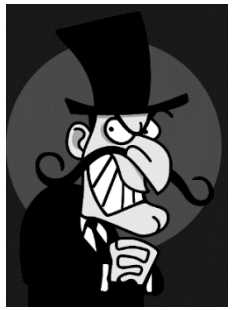


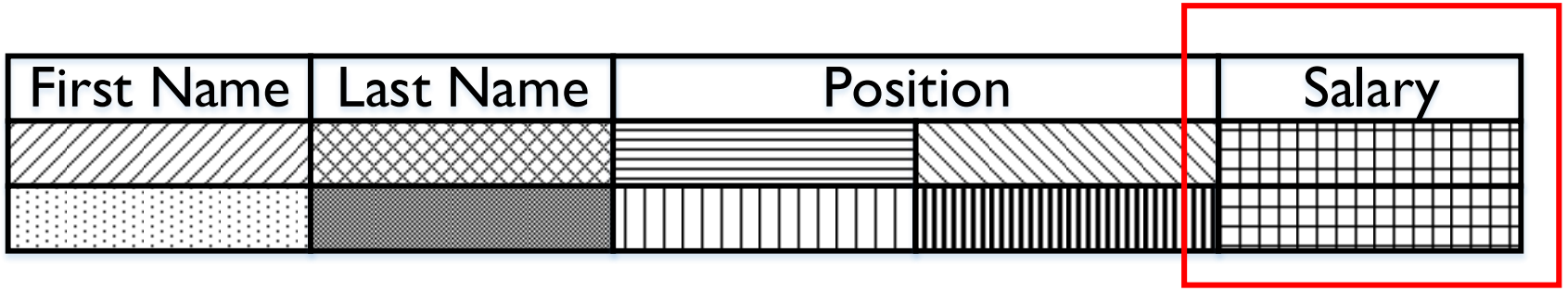
First Name	Last Name	Position	Salary

Rearranging Attack

HR

First Name	Last Name	Position	Salary
Adams	Wong	President	300,000
Hao	Yue	Assistant Professor	50,000



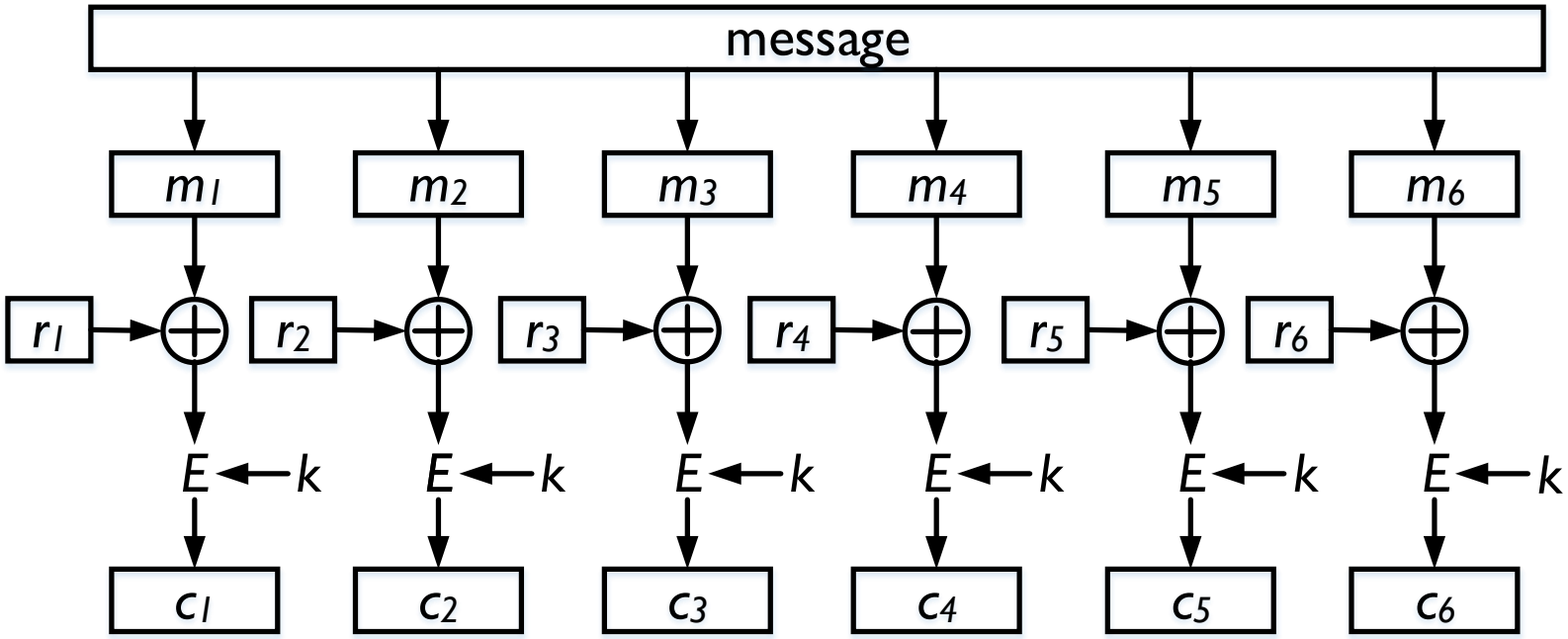


Payroll Office

First Name	Last Name	Position	Salary
Adams	Wong	President	300,000
Hao	Yue	Assistant Professor	300,000

Solution

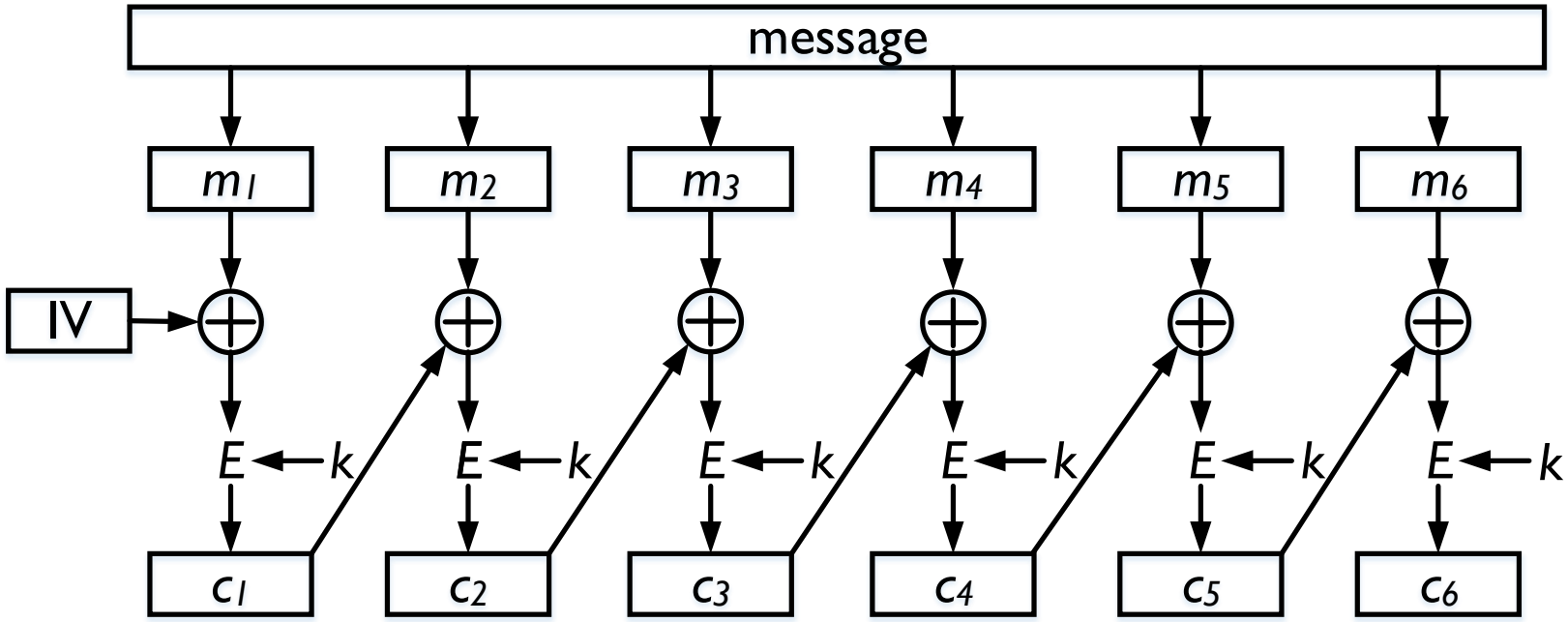
- XOR each block with a 64-bit random number before encrypting it with DES



- Drawback: the volume of data transmitted doubles (random number + ciphertext)

Cipher Block Chaining (CBC)

- Use the previous ciphertext block as random number and XOR it with the next plaintext block
- Select a random number (called initialization vector, IV) that is XORed with the first plaintext block. Why?



Pros and Cons

- Identical plaintext blocks will not cause repeats in the ciphertext
- IV needs to be shared between sender and receiver
- Error in one received ciphertext block will affect the correct decryption of next ciphertext block

Hash Algorithms

- Hash algorithm
  - Compression of data into a hash value, i.e., $y=h(m)$
  - Such algorithms are generally useful in systems (speed/space optimization)
- Must meet the following requirements to be used in cryptosystems
  - One-way (non-reversible): computationally hard to invert $h()$, i.e., compute $h^{-1}(y)$, where $y=h(m)$
  - Collision resistant: computationally hard to find two messages x_1 and x_2 such that $h(x_1)=h(x_2)$
- Question: What can you do with these constructs?

Birthday Paradox

- What is the minimum length of $h(m)$ to ensure collision resistant?
- Birthday Paradox
  - The probability that two or more people in a group of 23 share the same birthday is larger than 50%
- General Formulation
  - Given function f with n possible outputs that are uniformly distributed on k inputs $\{m_1, m_2, \ldots, m_k\}$.
    If $k > 1.2n^{1/2}$, $\Pr[f(m_i)=f(m_j)]>0.5$, for some i and j, $i \neq j$
  - Ex. $1.2\times(365^{1/2})\approx 23$

Birthday Paradox

- What is the minimum length of *h(m)* to ensure collision resistant?
- If the length of *h(m)* is *n*, it takes $O(2^{n/2})$ to find two messages with the same hash result
- The length of *h(m)* should be sufficient to resist the brute-force attacks based on birthday paradox

Message Authentication Code

- Authenticate the integrity of messages
  - Given hash function *h()*, key *k*, and message *m*

    MAC(k, m) = h(m|k)
  - Send both message *m* and the message authentication code *MAC(k, m)* to the receiver
  - The receiver computes *h(m|k)* using the received message and compares the result with the received *MAC(k, m)*
- Q: Why does *MAC(k, m)* provide integrity?
  - Cannot generate *MAC(k, m)* without knowing the key *k*
- Can we use h(m) instead?