Data Encryption Standard (DES)

密碼學與應用

海洋大學資訊工程系

丁培毅

DES

- Data Encryption Standard (Data Encryption Algorithm, DEA)
- 1973 National Bureau of Standards (NBS) RFP
- 1974 IBM LUCIFER
- National Security Agency (NSA) modified it
- 1977 NBS officially made it a standard
- major controversies:
  - 56-bit key size too short, $2^{56} \approx 7.2 \cdot 10^{16}$
  - NSA involvement (trapdoor?)

Note: A general purpose computer can do $2 \cdot 10^{9} \sim 2^{30.9}$ instructions/sec, there are 365*86400~ $2^{24.9}$ seconds/year i.e. $2^{55.8} \sim 6.4 \cdot 10^{16}$ instructions/year

NSA's Evil Claws

- NSA backdoors in RSA's BSafe library
  - Sept. 2013, RSA denied that "NSA has paid $10M to RSA to put the probably flawed Dual-EC-DRBG/NIST SP 800-90 as the default PRNG". Dual-EC-DRBG algorithm is fatally flawed, as Ferguson and Shumow pointed out in 2007.
  - Jan. 2014, Extended Random protocol (designed by NSA) to the discredited Dual Elliptic Curve random number generator - speed up the discovery of keys 65000 times
- Malware -- Regin
  - First appeared 2011, Nov. 2014 Kaspersky: The Regin platform: nation-state ownage of GSM networks
- Worldwide surveillance project: PRISM
  - June. 2013, Edward Snowden disclosed NSA's PRISM

DES

- Used extensively in computer network environments and electronic commerce
- Major Attacks:
  - Hardware DES crackers: 1977 Diffie and Hellman 1993 Wiener, 1997 Verser, 1998 EFF
  - 1990 Biham and Shamir, Differential Cryptanalysis
  - 1993 Masui, Linear Cryptanalysis
- Five-year reviews: 1982, 1987, 1992 passed (http://www.itl.nist.gov/fipspubs/fip46-2.htm) , 1997??
- Replacement: 2000 NIST AES (Rijndael)

Simplified DES-Type Algorithm

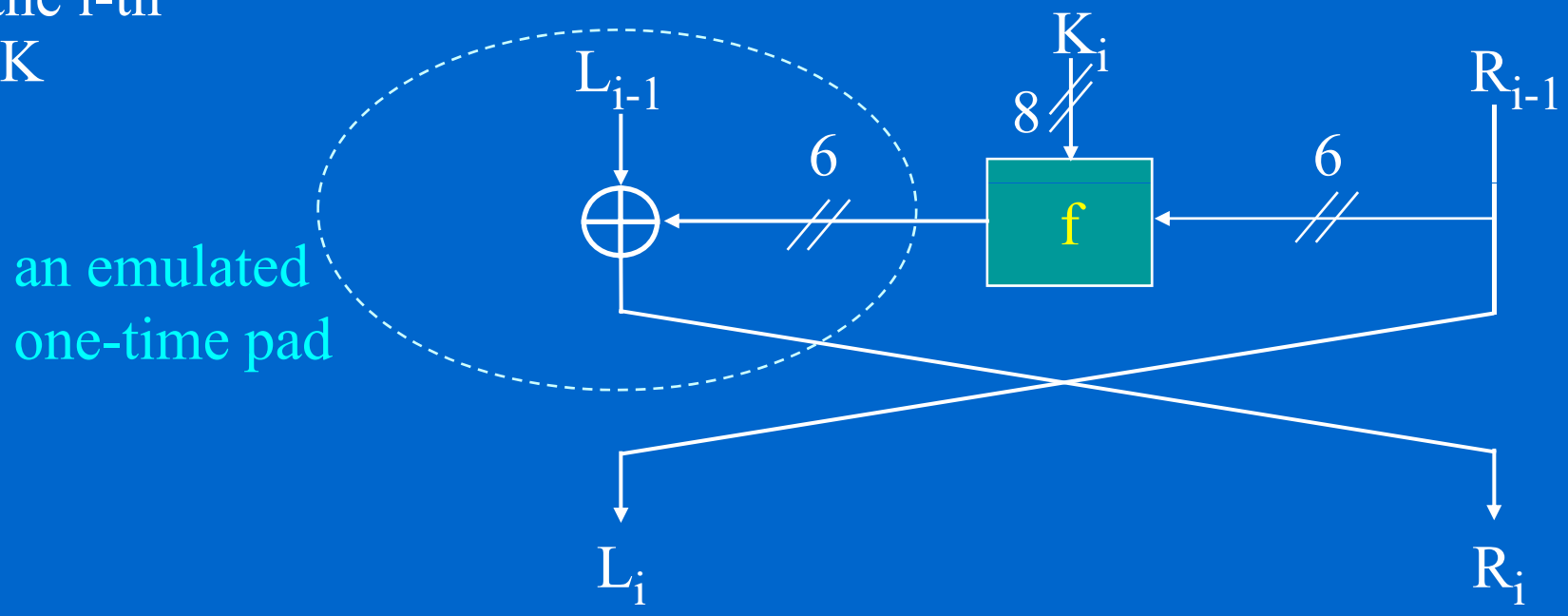
- A Block cipher:
  - 12-bit message, written in the form L_0R_0, each 6 bits
  - 9-bit key K
  - n rounds, each round converts $L_{i-1}R_{i-1}$ to L_iR_i using an 8-bit key K_i derived from K (starting from the i-th bit of K)
  - main part is a nonlinear round function $f(R_{i-1}, K_i)$ which is called a Feistel (1973 IBM LUCIFER) system, commonly used in many symmetric encryption schemes that maximize the effects of Shannon's "Confusion" and "Diffusion"

Feistel System

- $f(R_{i-1}, K_i)$ takes a 6-bit input R_{i-1} and an 8-bit input K_i, and produces a 6-bit output

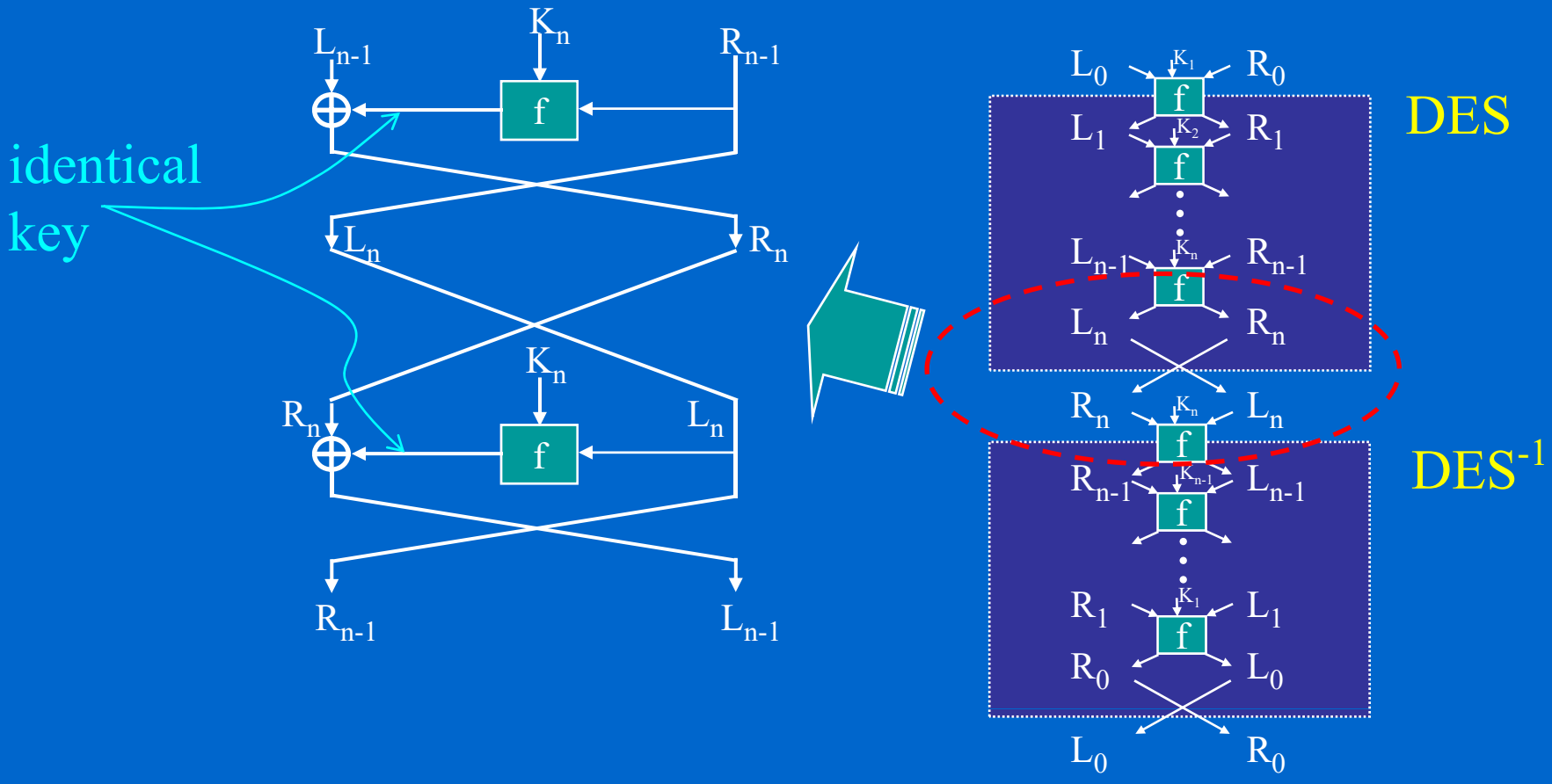
starting from the i-th bit of K

$$L_i = R_{i-1} \text{ and } R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$$



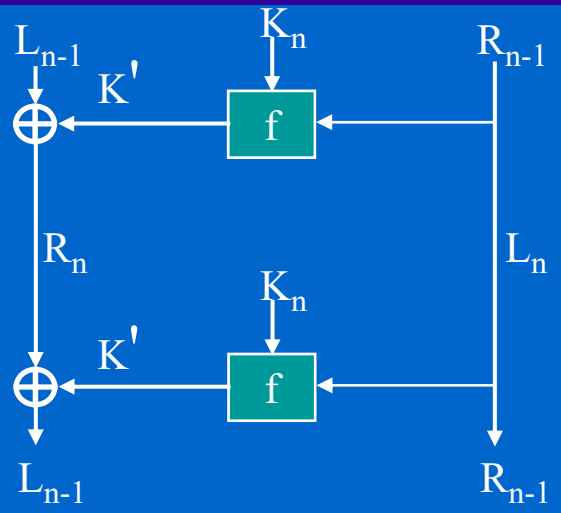
Feistel System

- How to encrypt/decrypt with a Feistel structure?



Feistel System

- Another view:



- Intuitively, f(·) should be designed s.t.

  1). output K' is not correlated to L_{n-1} or R_n;

  2). K' is as random (unpredictable) as possible;

  3). K' can not be reproduced from R_{n-1} (or L_n) without knowing K_n;

  4). given many pairs of (L_{n-1}, R_n, R_{n-1}), it should not be easy to deduce K_n (f(·) should behave like a one way function w.r.t input

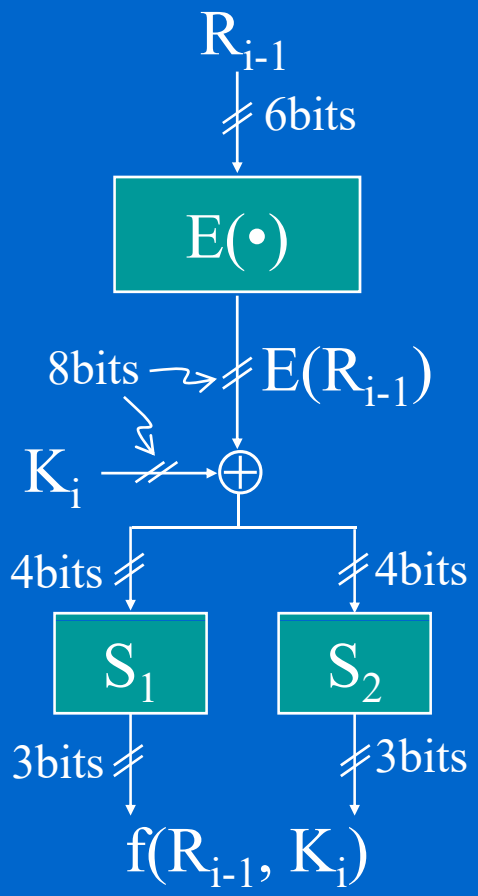
  K_n actually not possible for the limited bit length)

Feistel Type of Systems

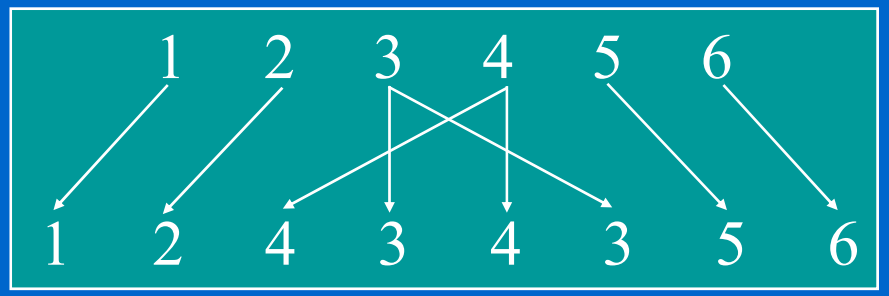
	Block Size	Key Size	# Rounds
DES	64	56	16
Double-DES	64	112	32
Triple-DES	64	168	48
IDEA	64	128	8
Blowfish	64	32..448	16
RC5	32, 64, 128	0..2048	variable
CAST-128	64	40..128	16
RC2	64	8..1024	16

Design of $f(R_{i-1}, K_i)$

- $f(R_{i-1}, K_i)$ provides an autokey stream for encrypting L_{i-1}



The expander function E(•)



S-boxes (Substitution-boxes)

S_1	101	010	001	110	011	100	111	000
	001	100	110	010	000	111	101	011

S_2	100	000	110	101	111	001	011	010
	101	011	000	111	110	010	001	100

Design of $f(R_{i-1}, K_i)$

- What happens if there were no $E(\cdot)$ and S_1, S_2?

  $K_i' = f(R_{i-1}, K_i) = R_{i-1} \oplus K_i$

  means that once you know a set of $L_{i-1}\, R_{i-1}\, R_i$ you know K_i

  $K_i = R_{i-1} \oplus L_{i-1} \oplus R_i$

  the overall DES output is then a linear function of inputs and keys K_i's, you can solve system of equations for K_i's if you have enough pairs of (plaintext, ciphertext)'s.

- What happens if S_1, S_2 are linear operator like 'division'?

  $S_1^{-1}(y)$ could be one of the 2 pre-images x of S_1, namely,
  $S_1^{-1}(y)=2\cdot y$ or $S_1^{-1}(y)=2\cdot y+1$,
  if $S_1^{-1}(y)=2\cdot y$ then $K_i^L = E(R_{i-1})^L \oplus 2\cdot(L_{i-1} \oplus R_i)^L$
  the overall DES output is still a linear function

- S_1, S_2 are transformations requiring table lookup, nonlinear

Differential Cryptanalysis

- Biham and Shamir, "Differential cryptanalysis of DES-like cryptosystems," Crypto90
- Probably were known to the designers of DES at IBM and NSA, Coppersmith.

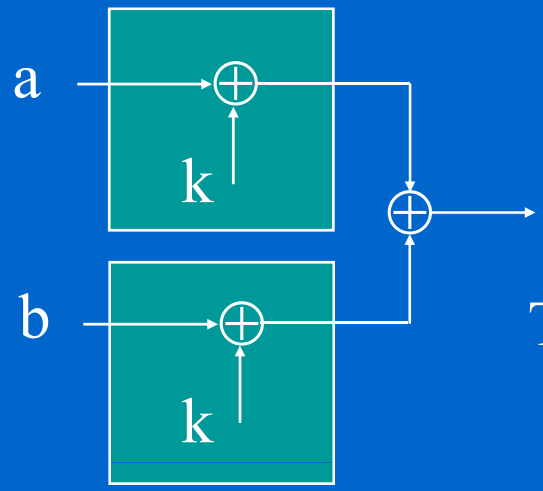
- Compare the differences in the plaintexts and the ciphertexts (XOR) for suitably chosen pairs of plaintexts and deduce information about the key.

  Note: XOR of two uniformly random bits should be uniformly random
  DES can be viewed as a PRNG, its output is close to random.

- Chosen plaintext attack: have access to an encryption engine

Differential Cryptanalysis

- Idea 1: The key is introduced into the system by XORing with $E(R_{i-1})$. It is possible to XOR two sets of outputs to remove the randomness effects introduced by the key.



The effects of k are removed.

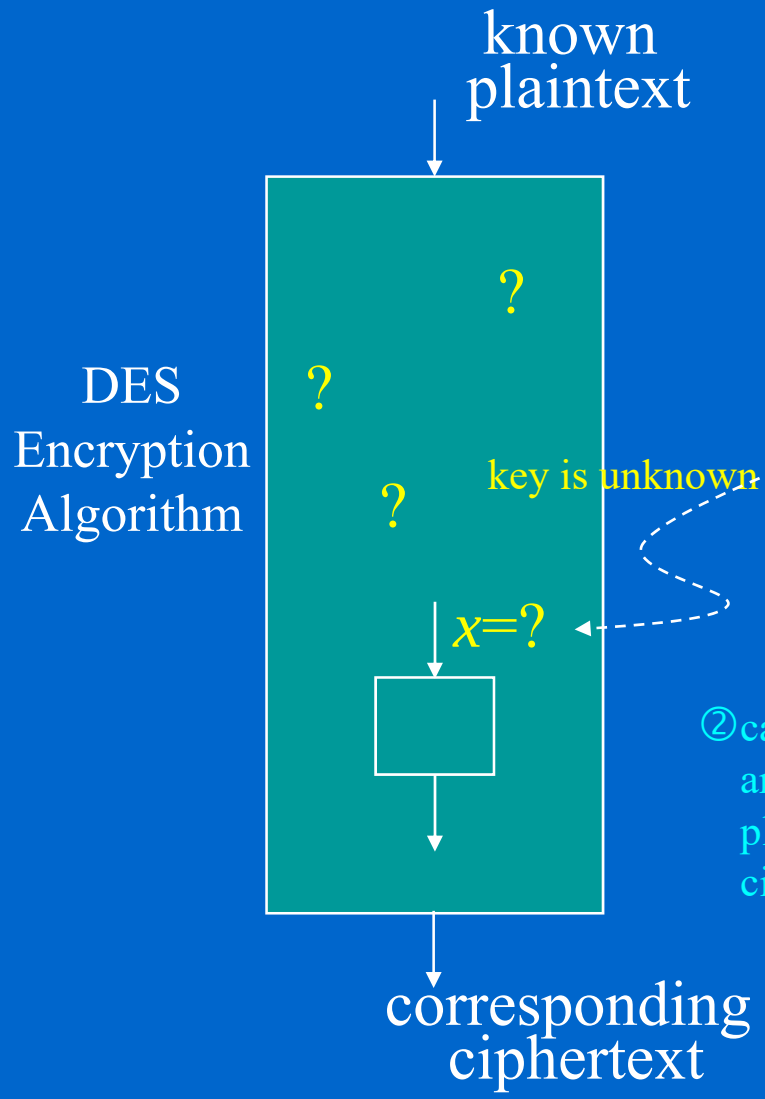
Differential Cryptanalysis

- Idea 2:
  - consider a nonlinear function $g(\cdot)$
  - inputs x_1, x_2 and outputs y_1, y_2 satisfy
    $y_1 = g(x_1)$, $y_2 = g(x_2)$
  - the XOR of inputs $x' = x_1 \oplus x_2$ and the XOR of outputs $y' = y_1 \oplus y_2$ are constrained also by $g(\cdot)$ such that given a pair of x' and y', there are only a few candidate pairs (x_1, x_2) satisfying the constraints
    - ex: given $x' (= x_1 \oplus x_2) = 010$, there are only 8 (out of 64) possible pairs of (x_1, x_2)

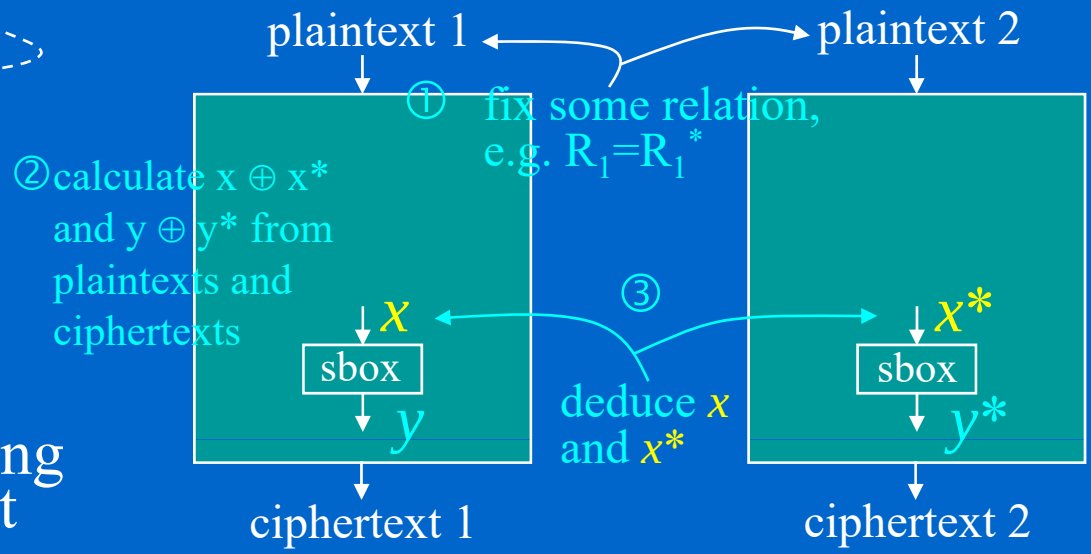
x_1	000	100	010	110	001	101	011	111
x_2	010	110	000	100	011	111	001	101

if $y' (= y_1 \oplus y_2)$ is also given as 101, try listing all $g(x_1)$, $g(x_2)$ and see which pairs of input (x_1, x_2) satisfy the constraint

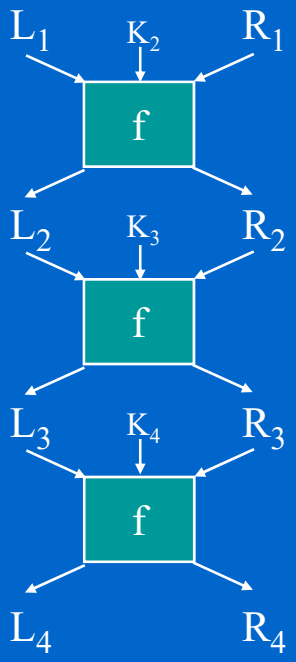
Differential Cryptanalysis



- Somewhere inside the algorithm, the key is XORed to the data stream. If we could deduce the value of some internal data *x*, we might be able to deduce the key.
- Is there any method we can use to get more specific about an unknown internal data ***x***?



3-Round Differential Cryptanalysis



$L_2 = R_1, R_2 = L_1 \oplus f(R_1, K_2)$

$L_3 = R_2, R_3 = L_2 \oplus f(R_2, K_3)$

$L_4 = R_3, R_4 = L_3 \oplus f(R_3, K_4)$
$= L_1 \oplus f(R_1, K_2) \oplus f(R_3, K_4)$

For another set of input $(L_1^*, R_1^* = R_1)$, the output is (L_4^*, R_4^*)

$R_4^* = L_1^* \oplus f(R_1^*, K_2) \oplus f(R_3^*, K_4)$

The difference $R_4' = R_4 \oplus R_4^* = L_1' \oplus f(L_4, K_4) \oplus f(L_4^*, K_4)$

no K_2, K_3 involved

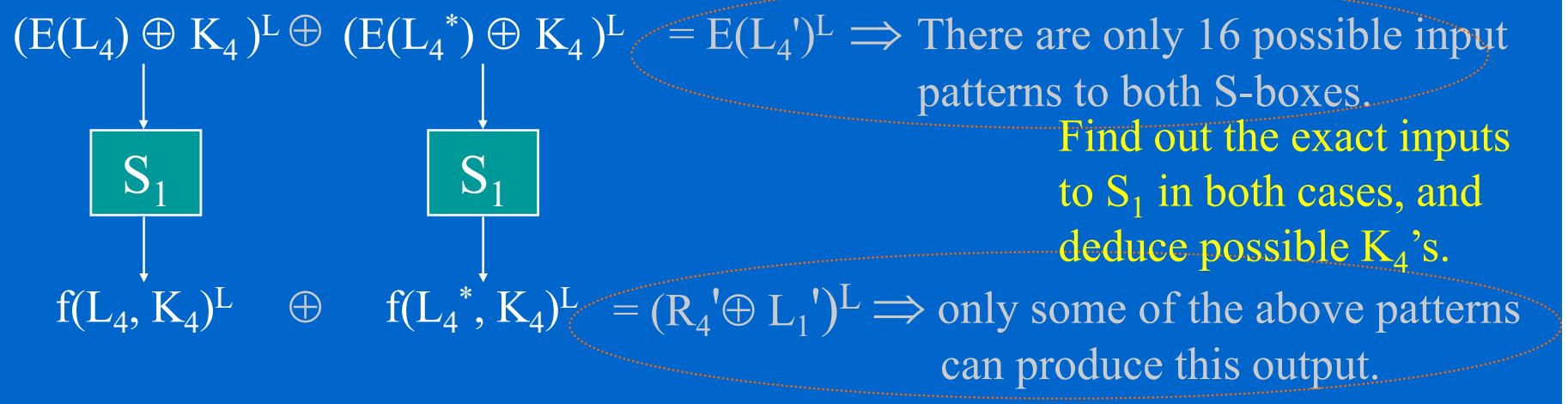
3-Round Differential Cryptanalysis

Rewrite as: $R_4' \oplus L_1' = f(L_4, K_4) \oplus f(L_4^*, K_4)$

If we know L_1, R_1, L_1^*, R_1^*, L_4, R_4, L_4^*, R_4^*, we know everything except K_4

To find out K_4, you can

1) try 256 combinations of K_4 in a brute-force manner or
2) find out suitable K_4, such that the input XOR to the S-box is $E(L_4')$ and the output XOR of the S-box is $R_4' \oplus L_1'$



3-Round Differential Cryptanalysis

- Ex. $L_4 = 101110$ $L_4^* = 000010$ Known fixed values

  Input XOR $E(L_4')^L = 1011$

  Output XOR $(R_4' \oplus L_1')^L = 100$

  – possible 16 input pairs: (1011, 0000) (1010, 0001)…

  – only (1010, 0001) and (0001, 1010) produce the specified output XOR 100

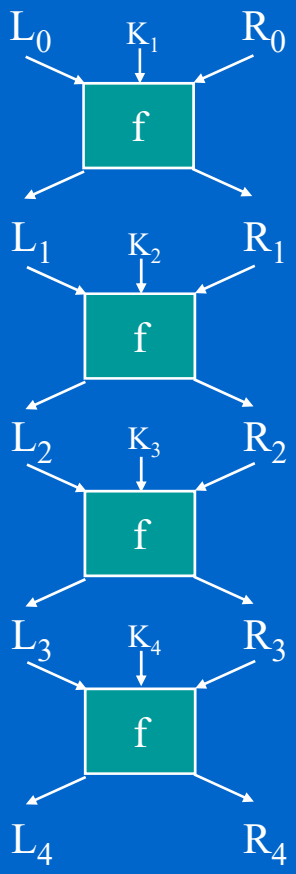
  – $(E(L_4) \oplus K_4)^L = 1011 \oplus K_4^L$ $(E(L_4^*) \oplus K_4)^L = 0000 \oplus K_4^L$

  – $K_4^L = 0001$ or 1010, repeat the procedure for some other data can eliminate one of them

  – K_4^R can be found by a similar procedure

  – guess the last bit of the key and verify on a (plaintext, ciphertext) pair

4-Round Differential Cryptanalysis



- Characteristics
  - chosen plaintext attack
  - know the complete algorithm except the key
  - probabilistic approach
- weakness in the S-box S_1: if we look at the 16 input pairs with XOR equal to 0011, we discover that 12 of them have output XOR equal to 011 (on the average, only 2 input pairs should yield a given output XOR)
- similar weakness in the S-box S_2: among the 16 input pairs with XOR equal to 1100, there are 8 output XOR equal to 010

4-Round Differential Cryptanalysis

- Ex. for S_1, XOR of input 0011, XOR of output:

S-box	S_1							
	101	010	001	110	011	100	111	000
	001	100	110	010	000	111	101	011

x_1	0000	0001	0010	0011	0100	0101	0110	0111
x_2	0011	0010	0001	0000	0111	0110	0101	0100
$y_1 \oplus y_2$	011	011	011	011	011	011	011	011
x_1	1000	1001	1010	1011	1100	1101	1110	1111
x_2	1011	1010	1001	1000	1111	1110	1101	1100
$y_1 \oplus y_2$	011	010	010	011	011	010	010	011

- The weakness of S-Boxes could waste our 3-round analysis more time, since a specified output XOR cannot eliminate as much input candidates.

4-Round Differential Cryptanalysis

- Idea: using the weakness of the S-boxes to create a good environment for 3-round cryptanalysis to work, i.e. choosing L_0, R_0 , L_0^*, R_0^* s.t. $R_1 = R_1^*$
- ex. $R_0' = 001100$ $\quad L_0' = 011010$ $\quad \begin{cases} L_0 = 000000 \\ L_0^* = 011010 \end{cases}$

$E(R_0') = 00111100$

$\Pr\{f(R_0, K_1) \oplus f(R_0^*, K_1) = 011010\} \approx 12/16*8/16=3/8$

over 64 possible pairs (R_0, R_0^*)

$E(R_0 \oplus R_0^*) = (E(R_0) \oplus K_1) \oplus (E(R_0^*) \oplus K_1)$

$$R_1' = R_1 \oplus R_1^* = (L_0 \oplus f(R_0, K_1)) \oplus (L_0^* \oplus f(R_0^*, K_1))$$
$$= L_0' \oplus (f(R_0, K_1) \oplus f(R_0^*, K_1))$$

$$\begin{cases} \Pr\{R_1' = 000000\} = 3/8 = \Pr\{R_1 = R_1^*\} \\ L_1' = R_0' = 001100 \end{cases}$$

3 out of 8 times we can apply 3-round analysis successfully to get K_4. If R_1' is not 000000, the derived keys are random. Try all 64 inputs, correct keys should appear more times.

4-Round Differential Cryptanalysis

- Ex. (continued)

K_4 First 4 bits	Frequency
0000	12
0001	7
0010	8
0011	15
0100	4
0101	3
0110	4
0111	6
1000	33
1001	40
1010	35
1011	35
1100	59
1101	32
1110	28
1111	39

K_4 Last 4 bits	Frequency
0000	14
0001	6
0010	42
0011	10
0100	27
0101	10
0110	8
0111	11
1000	8
1001	16
1010	8
1011	18
1100	8
1101	23
1110	6
1111	17

4-Round Differential Cryptanalysis

- Now that we know 8 out of 9 bits of the key K, i.e. $K = k_0k_1k_2{*}k_4k_5k_6k_7k_8k_9$, the last bit ($k_3$) can be guessed and verified by one (plaintext, ciphertext) pair.

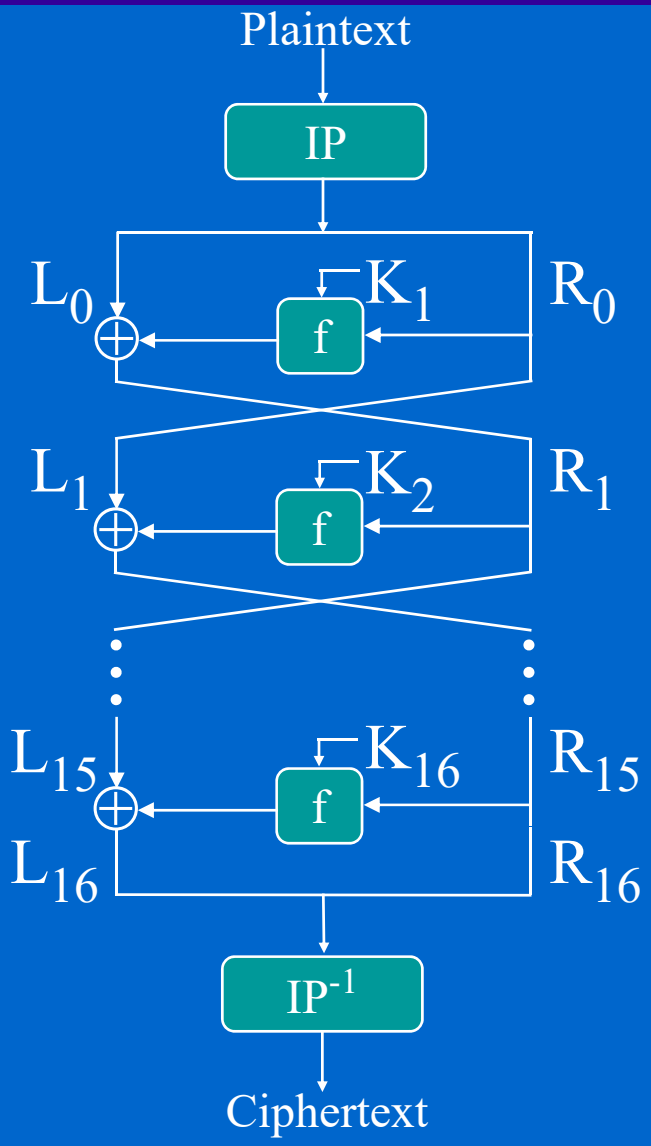
DES Design Criteria

- NBS suggested the following guidelines in 1973
  - High level of security
  - Complete specification and easy to understand
  - Security must be based on the key, not on the obscurity of the algorithm
  - System is available to all users
  - Easily adaptable for diverse applications
  - Economical implementation in electronic devices
  - Algorithm must be efficient to use
  - Algorithm must be easy to validate
  - Algorithm must be exportable

DES Algorithm

- A Block cipher:
  - 64-bit message, written in the form L_0R_0, each 32 bits
  - 56-bit key K, expressed as 64-bit string, 8-th, 16-th,... bits are parity bits
  - 16 rounds, each round converts $L_{i-1}R_{i-1}$ to L_iR_i using an 48-bit key K_i derived from K (with a key schedule)
  - main part is the nonlinear function $f(R_{i-1}, K_i)$

DES Algorithm



64 bit

64 → 64 permutation

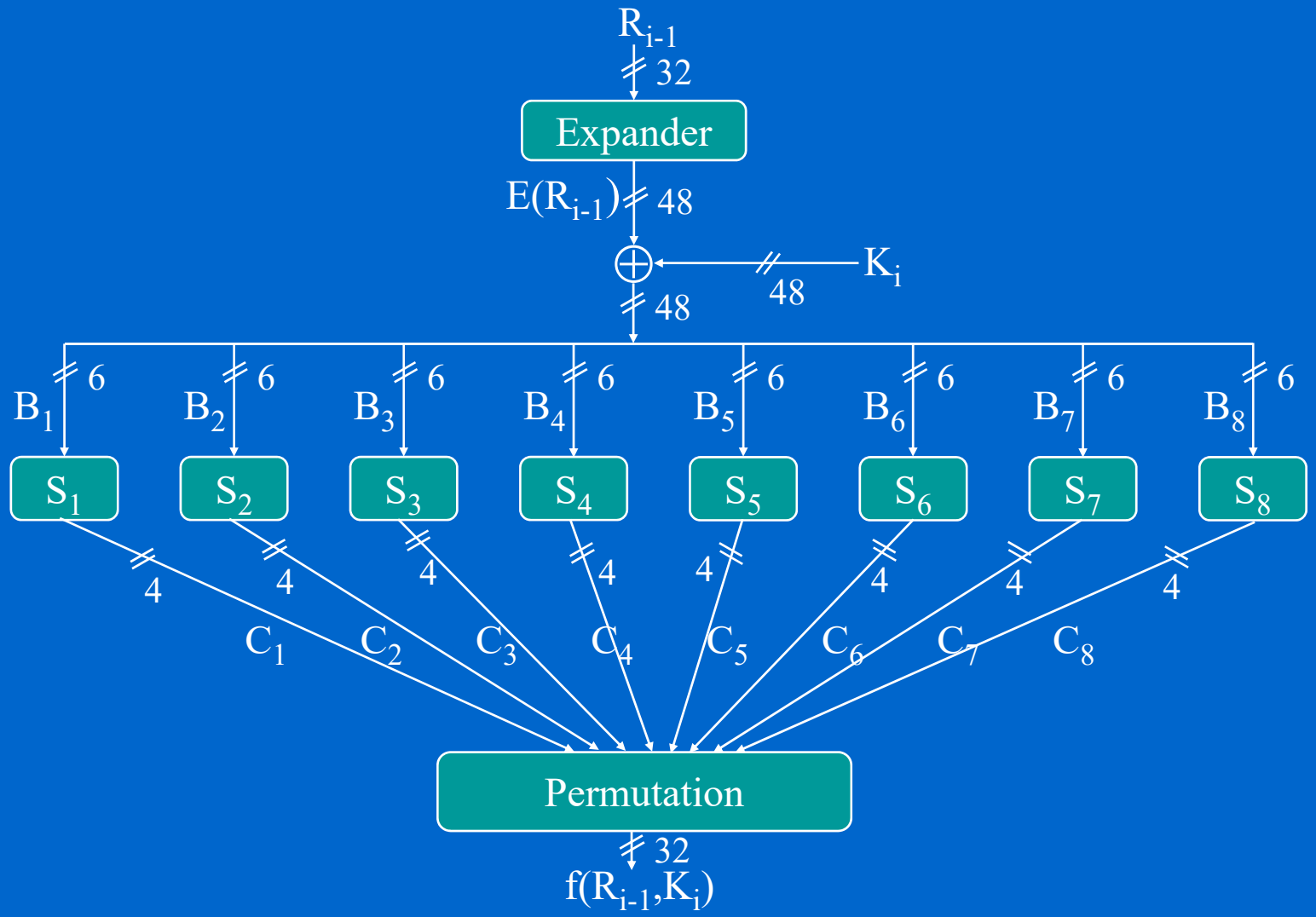
32 bit L_0, R_0, 48 bit K_1

32 bit L_{16}, R_{16}, 48 bit K_{16}

64 → 64 permutation

64 bit

DES Round Function $f(R_{i-1},K_i)$



Initial and Expansion Permutations

- Input: $b_1\ b_2\ b_3\ \ldots\ b_{32}\ b_{33}\ b_{34}\ b_{35}\ \ldots\ b_{64}$

MSB

- Initial Permutation (IP):

Input b_{58} becomes bit_1 i.e. MSB of L_0

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

- Expansion Permutation (EP):

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

b_{32} (LSB of R_{i-1}) becomes bit_1 (to be XORed to MSB of K_i)

S-Boxes Design Criteria

- IBM announced in 1990
  - Each S-box has 6 input bits and 4 output bits. This was the largest that could be put on one chip in 1974
  - The outputs of the S-boxes should not be close to being linear function of the inputs.
  - Each row of an S-box contains all numbers from 0 to 15
  - If two inputs to an S-box differ by 1 bit, the outputs must differ by 2 bits
  - If two inputs to an S-box differ in their first 2 bits but have the same last 2 bits, the outputs must be unequal.
  - There are 64 pairs of inputs having a given XOR. For each of these pairs, compute the XOR of the outputs (out of 16 possibilities) . No more than eight of these output XORs should be the same.

S-Boxes

S-box 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 2

15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S-box 3

10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S-box 4

7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S-Boxes

S-box 5

2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S-box 6

12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S-box 7

4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S-box 8

13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Permutation after S-Box

- 32 bit permutation: OP

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

Strong Diffusion Property of DES

- Small changes in plaintext or key cause significant changes in ciphertext (avalanche effect) … the expander cause this effect
  - Experiment: two plaintexts differ on one bit
    0000…..0 and 1000…..0
    and using the key
    00000011001011010010011000 10
    00111000011000001110001100 10

Round #	# of Bits that differ
0	1
4	39
8	29
12	30
16	34

Key Schedule

1. Parity bits are discarded
2. Permutation

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

3. Partition the results into C_0 D_0 each has 28 bits
4. $C_i = LS_i(C_{i-1})$ $D_i = LS_i(D_{i-1})$

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
LS_i	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

5. 48 bits K_i are chosen from the 56-bit string C_iD_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
41	52	31	37	47	55	30	40	51	45	33	48
44	49	39	56	34	53	46	42	50	36	29	32

Note: each bit of the 56 bit key is used in approximately 14 of the 16 rounds

Linear Cryptanalysis

- M. Matsui, "Linear cryptanalysis method for DES cipher," Eurocrypt'93
- A kind of "known plaintext attack"
- Break 8-round DES with 2^{21} known plaintexts
- Break 16-round DES with 2^{47} known plaintexts
- Break 8-round DES on natural English with 2^{29} ciphertext

  Note: $2^{55} \rightarrow 2^{47}$ 256 times saving (1 year $\rightarrow$ 1.4 days)

- Idea: try to find effective linear approximate equations

$$P[i_1,i_2,\ldots,i_a] \oplus C[j_1,j_2,\ldots,j_b] = K[k_1,k_2,\ldots,k_c]$$

XOR of *a* plaintext bits

such that they hold with probability $p \neq 1/2$ for randomly chosen plaintext.

Linear Cryptanalysis

- Once we have a linear approximate equation, it is possible to determine the bit $K[k_1, k_2, \ldots, k_c]$ through random experiments:
  - Assume $p > 1/2$
  - Obtain N random (plaintext, ciphertext) pairs
  - Let T be the number of plaintexts such that the expression $P[i_1, i_2, \ldots, i_a] \oplus C[j_1, j_2, \ldots, j_b] = 0$
  - if $T > N/2$ then guess $K[k_1, k_2, \ldots, k_c] = 0$
    else guess $K[k_1, k_2, \ldots, k_c] = 1$

Note: as N increases or $|p-1/2|$ increases, the accuracy of the guess increases

Linear Approximation of S-boxes

- For a given S-Box S_a (a=1,2,…,8), $1 \le \alpha \le 63$, $1 \le \beta \le 15$, define $NS_a(\alpha, \beta)$ as the number of input patterns (total 64 input *x* patterns) such that

$$x[1]\alpha[1] \oplus x[2]\alpha[2] \oplus x[3]\alpha[3] \oplus x[4]\alpha[4] \oplus x[5]\alpha[5] \oplus x[6]\alpha[6]$$
$$= S_a(x)[1]\beta[1] \oplus S_a(x)[2]\beta[2] \oplus S_a(x)[3]\beta[3] \oplus S_a(x)[4]\beta[4]$$

estimate of the probability that "a masked XOR value of the input bits coincides with a masked XOR value of the output bits"

- Ex. From the table on the next slide, $NS_5(16, 15) = 12$

$$\Rightarrow (R_{i-1}[17] \oplus K_i[26]) \oplus f(R_{i-1},K_i)[3,8,14,25] = 0$$

the 2nd input bit to S_5 — output from S_5

the 17-th bit of R_{i-1} where MSB is the 1st bit, i.e. $R_{i-1}[1]$

with p = 12/64=0.19

Linear Approximation of S-Box S_5

$NS_5(\alpha, \beta)-32$

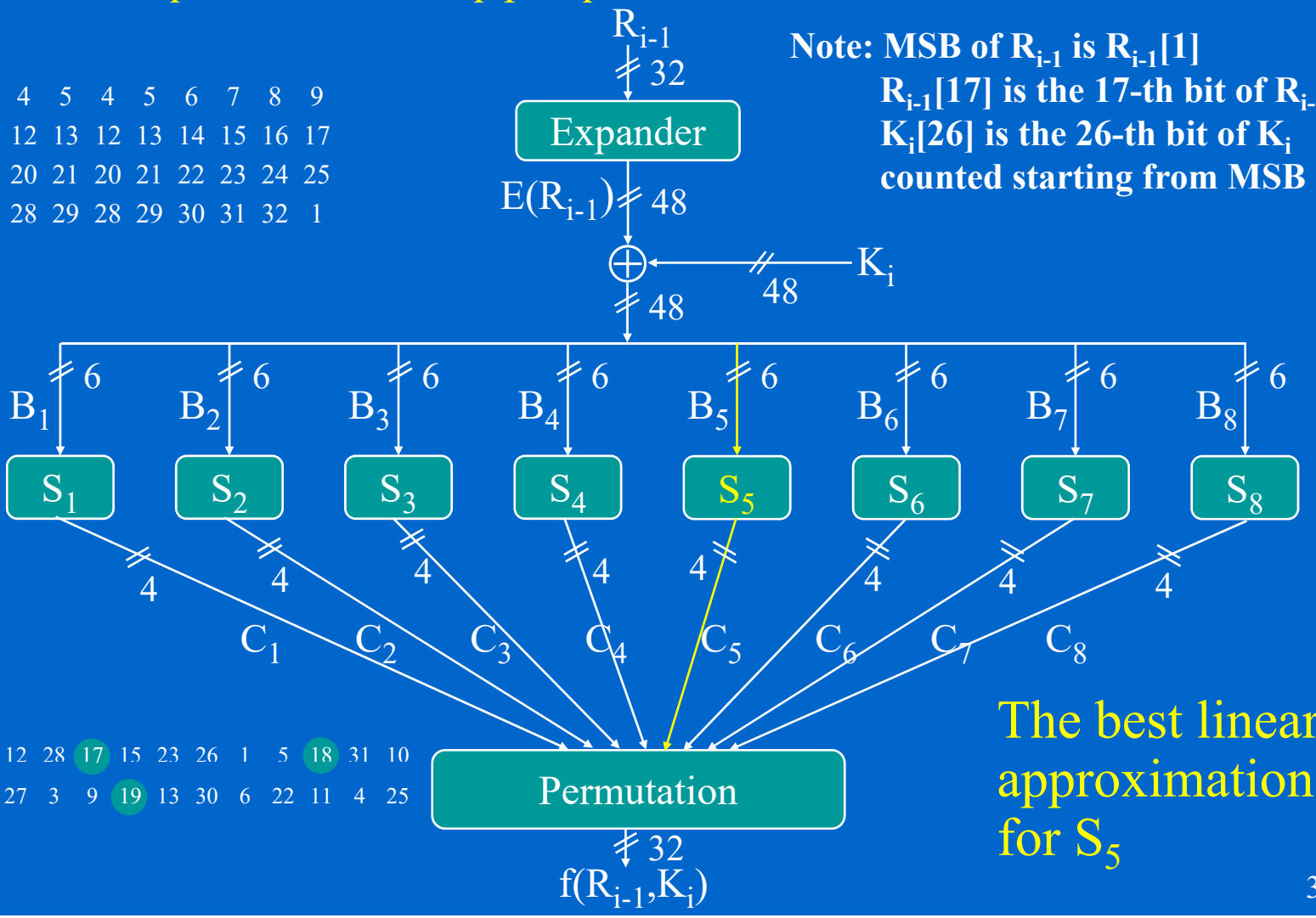
α \ β	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	4	-2	2	-2	2	-4	0	4	0	2	-2	2	-2	0	-4
3	0	-2	6	-2	-2	4	-4	0	0	-2	6	-2	-2	4	-4
4	2	-2	0	0	2	-2	0	0	2	2	4	-4	-2	-2	0
5	2	2	-4	0	10	-6	-4	0	2	-10	0	4	-2	2	4
6	-2	-4	-6	-2	-4	2	0	0	-2	0	-2	-6	-8	2	0
7	2	0	2	-2	8	6	0	-4	6	0	-6	-2	0	-6	-4
8	0	2	6	0	0	-2	-6	-2	2	4	-12	2	6	-4	4
9	-4	6	-2	0	-4	-6	-6	6	-2	0	-4	2	-6	-8	-4
10	4	0	0	-2	-6	2	2	2	2	-2	2	4	-4	-4	0
11	4	4	4	6	2	-2	-2	-2	-2	-2	2	0	-8	-4	0
12	2	0	-2	0	2	4	10	-2	4	-2	-8	-2	4	-6	-4
13	6	0	2	0	-2	4	-10	-2	0	-2	4	-2	8	-6	0
14	-2	-2	0	-2	4	0	2	-2	0	4	2	-4	6	-2	-4
15	-2	-2	8	6	4	0	2	2	4	8	-2	8	-6	2	0
16	2	-2	0	0	-2	-6	-8	0	-2	-2	-4	0	2	10	-20
17	2	-2	0	4	2	-2	-4	4	2	2	0	-8	-6	2	4
18	-2	0	-2	2	-4	-2	-8	4	6	4	6	-2	4	-6	0
19	-6	0	2	-2	4	2	0	4	-6	4	2	-6	4	-2	0

Linear Approximation of S-Box S_5

$$(R_{i-1}[17] \oplus K_i[26]) \oplus f(R_{i-1},K_i)[3,8,14,25] = 0$$

EP

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



Note: MSB of R_{i-1} is $R_{i-1}[1]$
$R_{i-1}[17]$ is the 17-th bit of R_{i-1}
$K_i[26]$ is the 26-th bit of K_i counted starting from MSB

OP

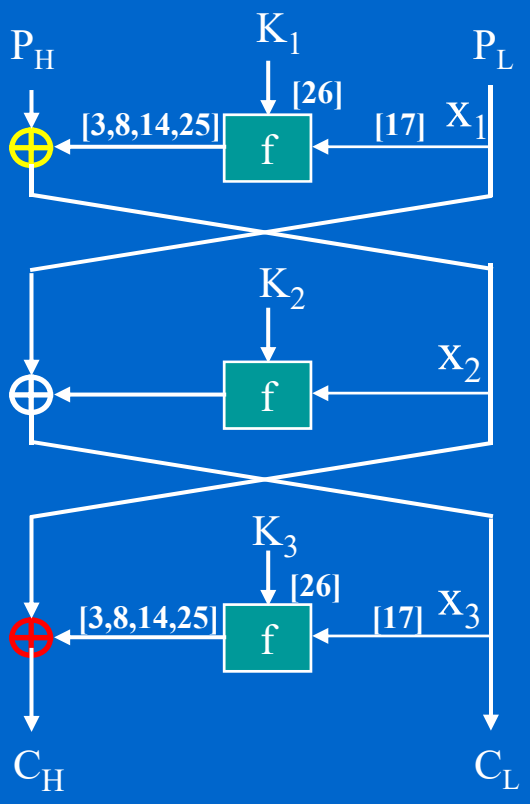
16	7	**20**	21	29	12	28	**17**	15	23	26	1	5	**18**	31	10
2	8	24	14	32	27	3	9	**19**	13	30	6	22	11	4	25

The best linear approximation for S_5

Linear Approximation of 3-round DES

- Extended approximation for S-Box to 3-round DES

$$(P_L[17] \oplus K_1[26]) \oplus P_H[3,8,14,25] = x_2[3,8,14,25]$$
$$(C_L[17] \oplus K_3[26]) \oplus x_2[3,8,14,25] = C_H[3,8,14,25]$$
$$\Downarrow \text{ (canceling common terms)}$$
$$P_H[3,8,14,25] \oplus C_H[3,8,14,25] \oplus P_L[17] \oplus C_L[17] = K_1[26] \oplus K_3[26]$$

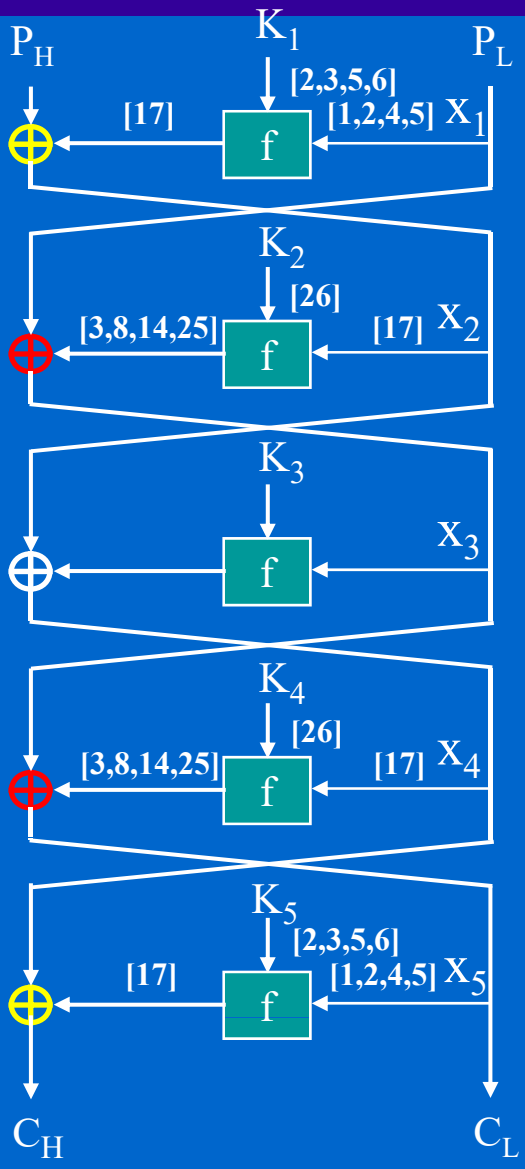


$$\Pr\{\text{the above eq. holds} \mid P, C\} = (12/64)^2 + (1-12/64)^2 = 0.6953$$

equal at the same time; unequal at the same time

This is the best linear approximation of 3-round DES cipher. Using this equation we can deduce $K_1[26] \oplus K_3[26]$

Linear Approximation of 5-round DES



Linear Approximation of 5-round DES

- For a 5-round DES, we can apply the previous linear approximation to the 2nd and the 4th round:

$$(x_2[18] \oplus K_2[26]) \oplus P_L[3,8,14,25] = x_3[3,8,14,25]$$
$$(x_4[18] \oplus K_4[26]) \oplus x_3[3,8,14,25] = C_L[3,8,14,25]$$

And apply the following linear equation (deduced from $NS_1(27, 4) = 22$)

$$X[1,2,4,5] \oplus f(X,K)[18] = K[2,3,5,6]$$

to the 1st and the 5th round:

$$P_L[1,2,4,5] \oplus K_1[2,3,5,6] = P_H[18] \oplus x_2[18]$$
$$x_5[1,2,4,5] \oplus K_5[2,3,5,6] = x_4[18] \oplus C_H[18]$$

combining these four linear equations (canceling common terms)

$$P_H[18] \oplus P_L[1,2,3,4,5,8,14,25] \oplus C_H[18] \oplus C_L[1,2,3,4,5,8,14,25] = K_1[2,3,5,6] \oplus K_2[26] \oplus K_4[26] \oplus K_5[2,3,5,6]$$

the success probability is shown to be 0.519

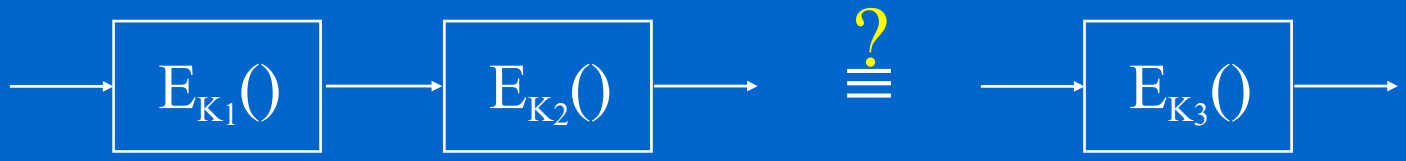
Best Expressions for DES cipher

round	Best approximation expressions	Success probability
3	$P_H[3,8,14,25] \oplus C_H[3,8,14,25] \oplus P_L[18] \oplus C_L[18] = K_1[26] \oplus K_3[26]$	$0.5+1.56\times2^{-3}$
4	$P_H[3,8,14,25] \oplus C_H[18] \oplus P_L[18] \oplus C_L[1,2,4,5,3,8,14,25] = K_1[26] \oplus K_3[26] \oplus K_4[2,3,5,6]$	$0.5-1.95\times2^{-5}$
5	$P_H[18] \oplus P_L[1,2,4,5,3,8,14,25] \oplus C_H[18] \oplus C_L[1,2,4,5,3,8,14,25] = K_1[2,3,5,6] \oplus K_2[26] \oplus K_4[26] \oplus K_5[2,3,5,6]$	$0.5-1.22\times2^{-6}$
⋮		
16	$P_H[8,14,25] \oplus P_L[16,20] \oplus C_H[17] \oplus C_L[1,2,4,5,3,8,14,25] = K_1[25,29] \oplus K_3[26] \oplus K_4[4] \oplus K_5[26] \oplus K_7[26] \oplus K_8[4] \oplus K_9[26] \oplus K_{11}[26] \oplus K_{12}[4] \oplus K_{13}[26] \oplus K_{15}[26] \oplus K_{16}[2,3,5,6]$	$0.5-1.49\times2^{-24}$
⋮		

? 2 key bits of 16-round DES can be deduced with high success rate using $|1.49 \times 2^{-24}|^{-2} \approx 2^{47}$ known-plaintexts

Is DES a group?

- Is DES closed under composition?
- $\forall x, E_{K_2}(E_{K_1}(x)) \stackrel{?}{\equiv} E_{K_3}(x)$



- If it were the case, double encryption with DES is no more secure than a single DES.

Is DES a group?

- Campbell and Wiener, "DES is not a group," Crypto92, pp.512-520.
- Fact: E_0 (DES encryption with K='000…0')
  E_1 (DES encryption with K='111…1')
  repeatedly apply E_1E_0 on a certain P yielded P after many iterations, i.e. $(E_1E_0)^n(P) = P$, where n is the smallest positive integer
- Lemma: If m is the smallest positive integer such that $(E_1E_0)^m(P) = P$ for all P, and n is the smallest positive integer such that $(E_1E_0)^n(P_0) = P_0$ for a particular P_0, then n divides m
  - proof: Let $m = nq + r$, $0 \le r < n$, since $(E_1E_0)^n(P_0) = P_0$
    $P_0 = (E_1E_0)^m(P_0) = (E_1E_0)^r(E_1E_0)^{nq}(P_0) = (E_1E_0)^r(P_0)$
    since n is the smallest integer s.t. $(E_1E_0)^n(P_0) = P_0 \Rightarrow r = 0$

Is DES a group?

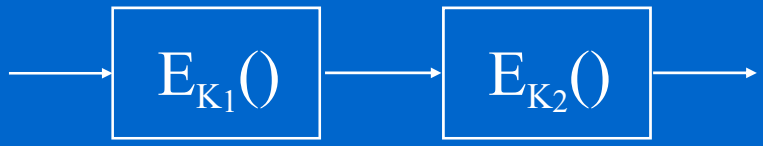
- Suppose DES is closed,
  - $E_1E_0 = E_K$
  - E_K^2, E_K^3,… are also represented by DES keys
  - there are only 2^{56} possible keys, consider the set $\{E_K, E_K^2, E_K^3 \ldots E_K^{2^{56}+1}\} \Rightarrow \exists\ i, j\ ,\ 1 \le i < j \le 2^{56}+1$, s.t. $E_K^j = E_K^i$
  - for any x, encrypt j times then decrypt i times yields x
    ie. $E_K^{j-i} = D_K^i E_K^j = D_K^i E_K^i = I$
  - therefore, the smallest m such that $(E_1E_0)^m(P) = P$ for all P, m must be less than or equal to 2^{56}
- Coppersmith found 33 P_0 and their corresponding n such that $(E_1E_0)^n(P_0) = P_0$. From the lemma, m must be the least common multiple of these n's, which turned out to be around 10^{277}. Contradiction! DES is not a group!

Double DES

- Fortunately, DES is not a group; it makes some sense to encrypt twice in the hope to increase the security of the system, however, double DES is still considered not much more secure than single DES.

- Meet in the middle attack reduce the strength of double DES from a seemingly 112-bit security to just 57-bit security under some assumptions

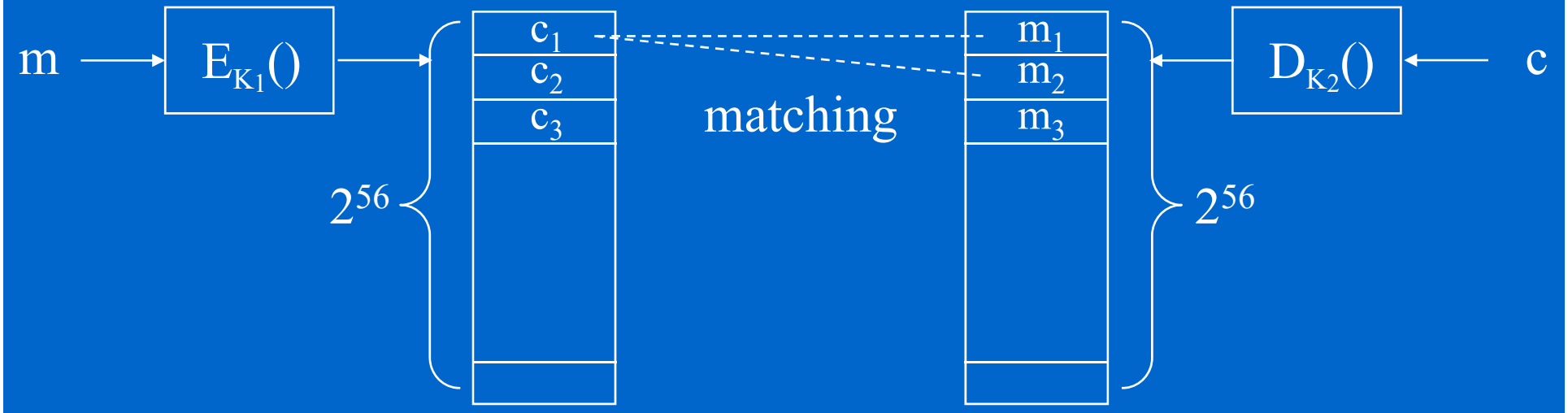
Brute-Force Attack



- A naïve brute-force attack can try all possible combinations of 2^{56} keys for K_1 and 2^{56} keys for K_2
- Number of all possible different keys for (K_1, K_2) would be 2^{112}
- The attacker has to do 2^{112} DES to exhaust all possible combinations

Meet-In-The-Middle Attack

- Given a pair of plaintext and ciphertext (m, c), try to find the key pair K_1, K_2 of a double DES scheme in a brute-force way



Computation Speed

- Software: ≈10 Mbits/sec

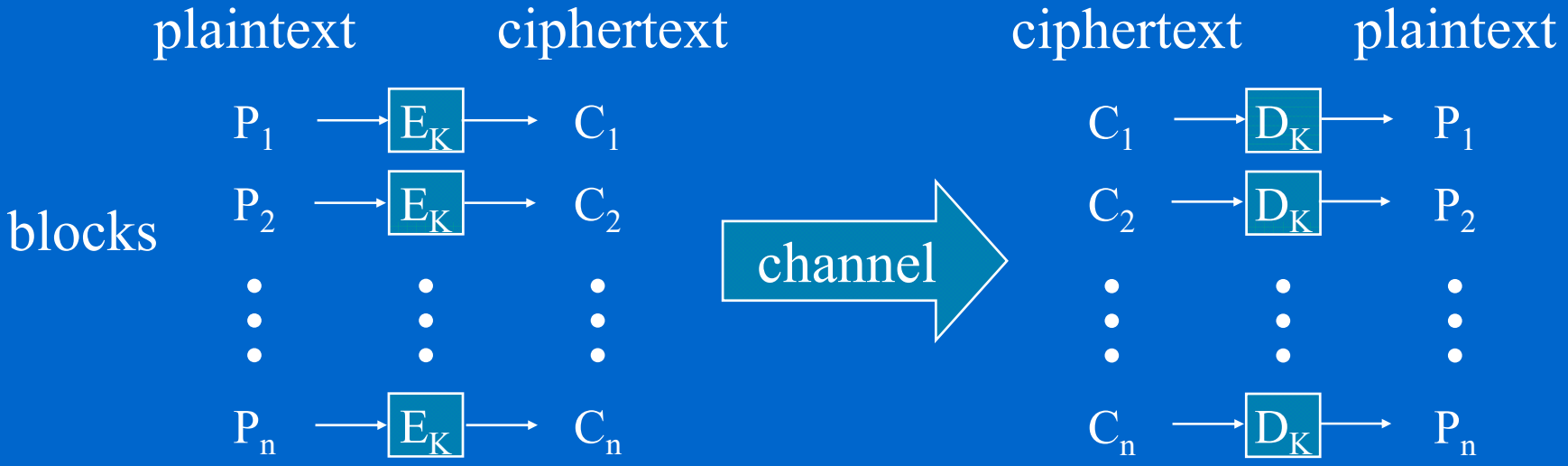
  ex. 1000 blocks/sec PIII 800 crypto++

- Hardware: > 1Gbits/sec

Mode of Operation

- Block cipher
  - ECB
  - CBC
- Stream cipher
  - Self-synchronizing stream cipher
    - CFB
  - Synchronous stream cipher
    - OFB
    - Counter Mode

Electronic CodeBook (ECB) Mode



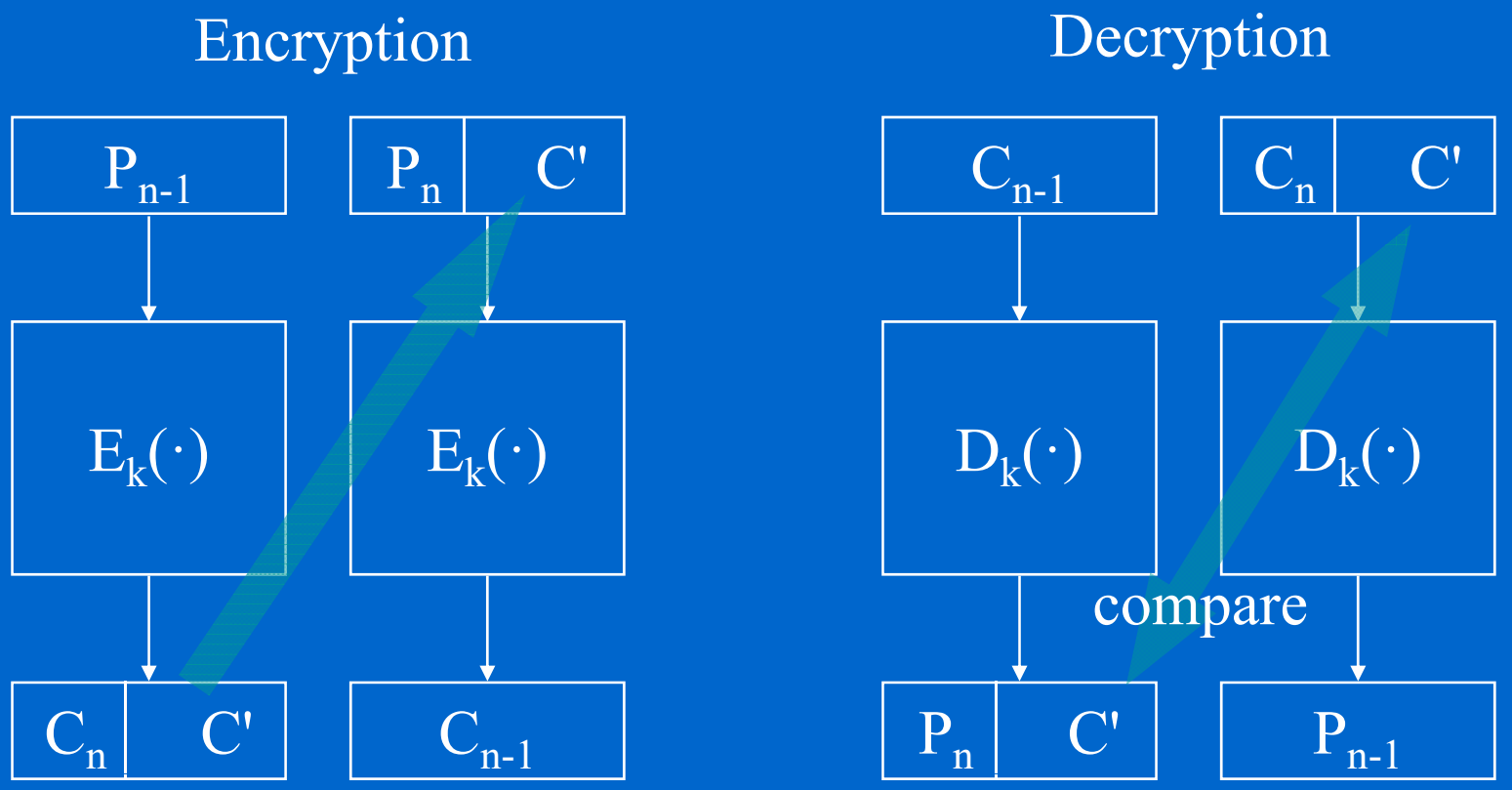
- The most obvious way to use a block cipher
- The same block of plaintext always map to the same cipher block (in the codebook).
- Vulnerability is greatest at the beginning and end of messages (easy to locate), or other well formatted sessions.

Electronic CodeBook (ECB) Mode

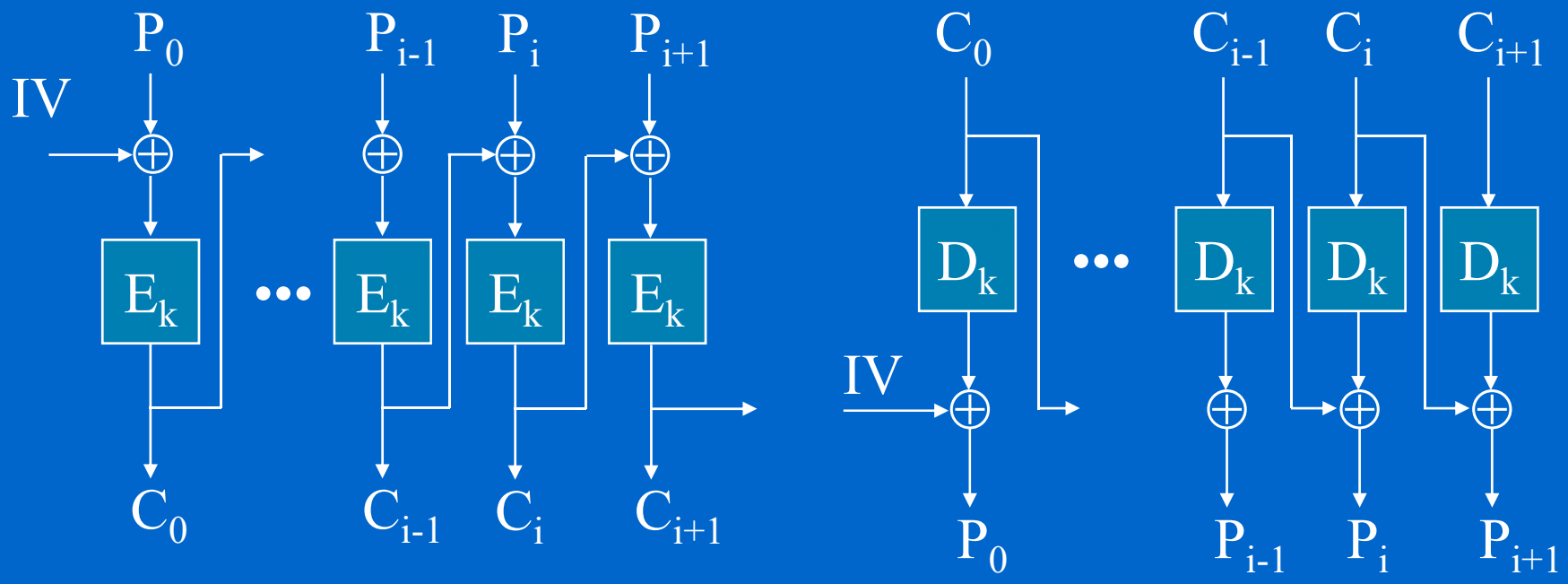
- Padding is necessary: fixed pattern of bits or ciphertext stealing.
- If the message have many blocks (encrypted with the same key) and some of the plaintexts are known, Eve can compile a codebook to map useful pieces of plaintext and ciphertext. She can effectively decipher the content of the message or alter the message without knowing the actual key.
- Eve can use block-replaying attack to alter part of the content of an important transaction.

Ciphertext stealing in ECB mode

- Take part of the last ciphertext block as the padding

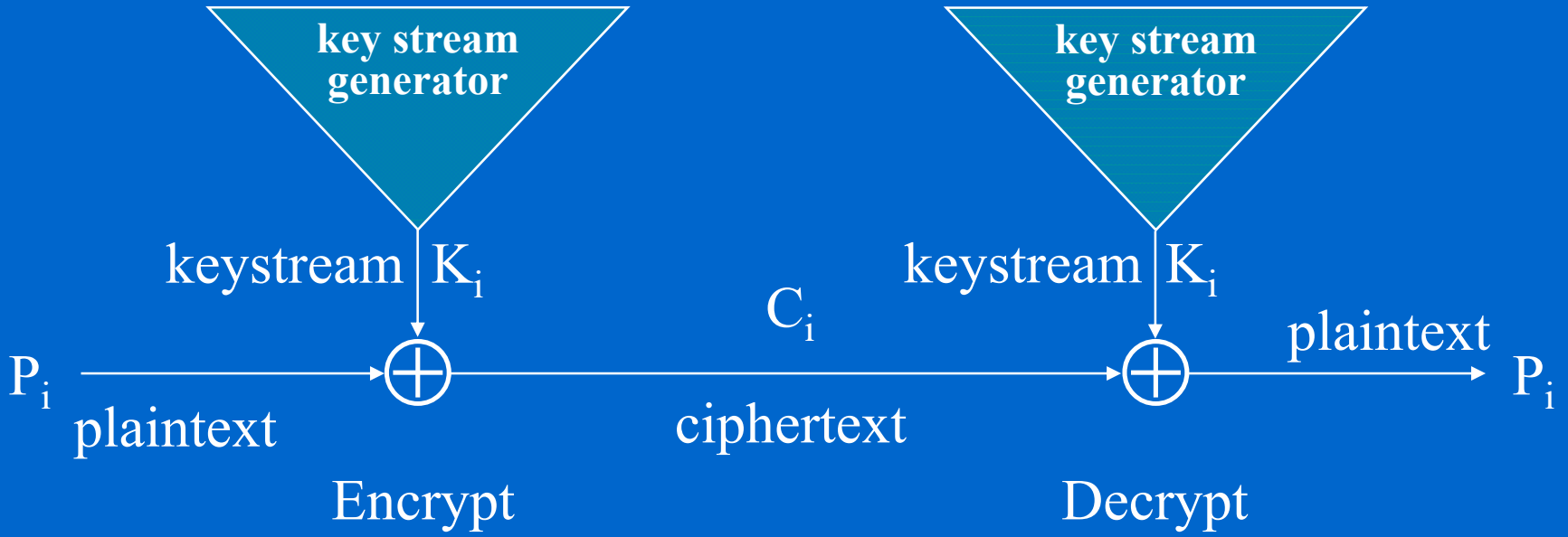


Cipher Block Chaining Mode



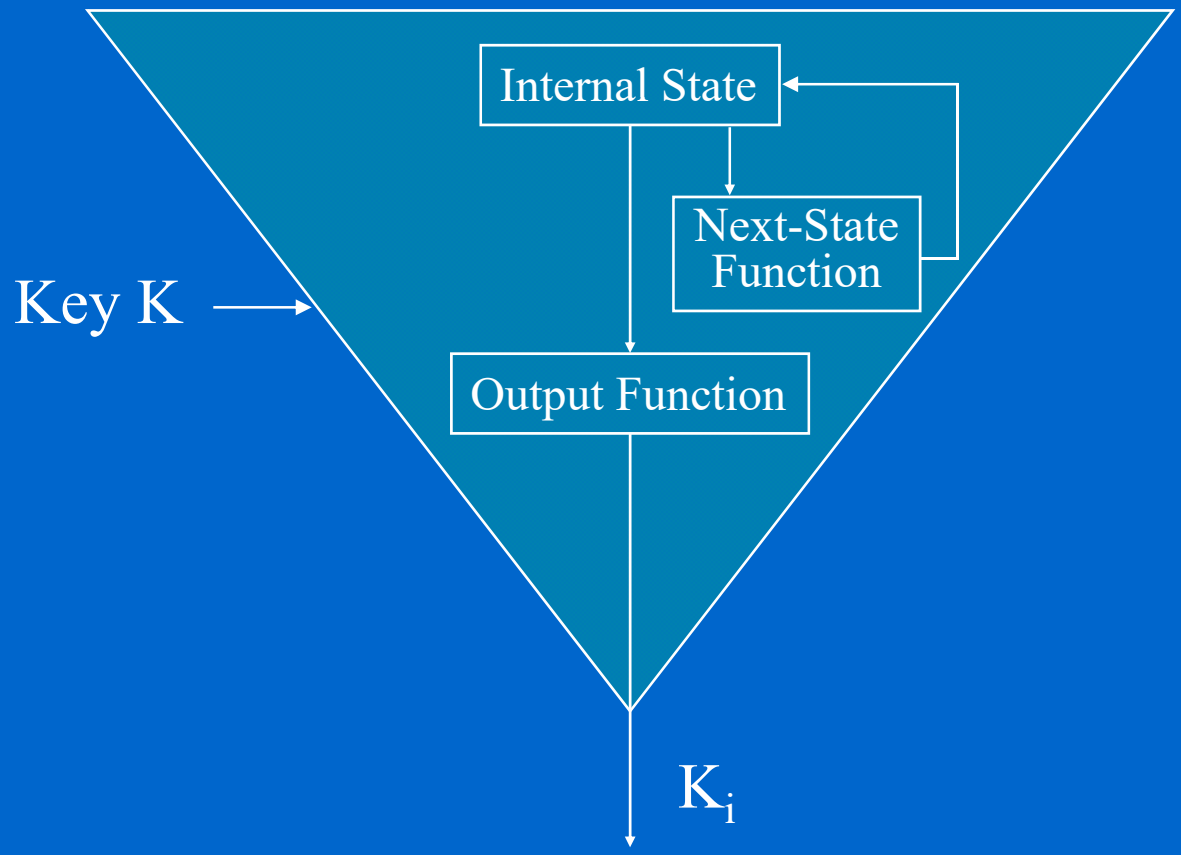
- A single-bit error in the ciphertext affects one block and one bit of the recovered plaintext.
- Self-recovering from ciphertext error. It doesn't recover at all from synchronization errors. (one bit is added or lost in the ciphertext stream)

Stream Cipher



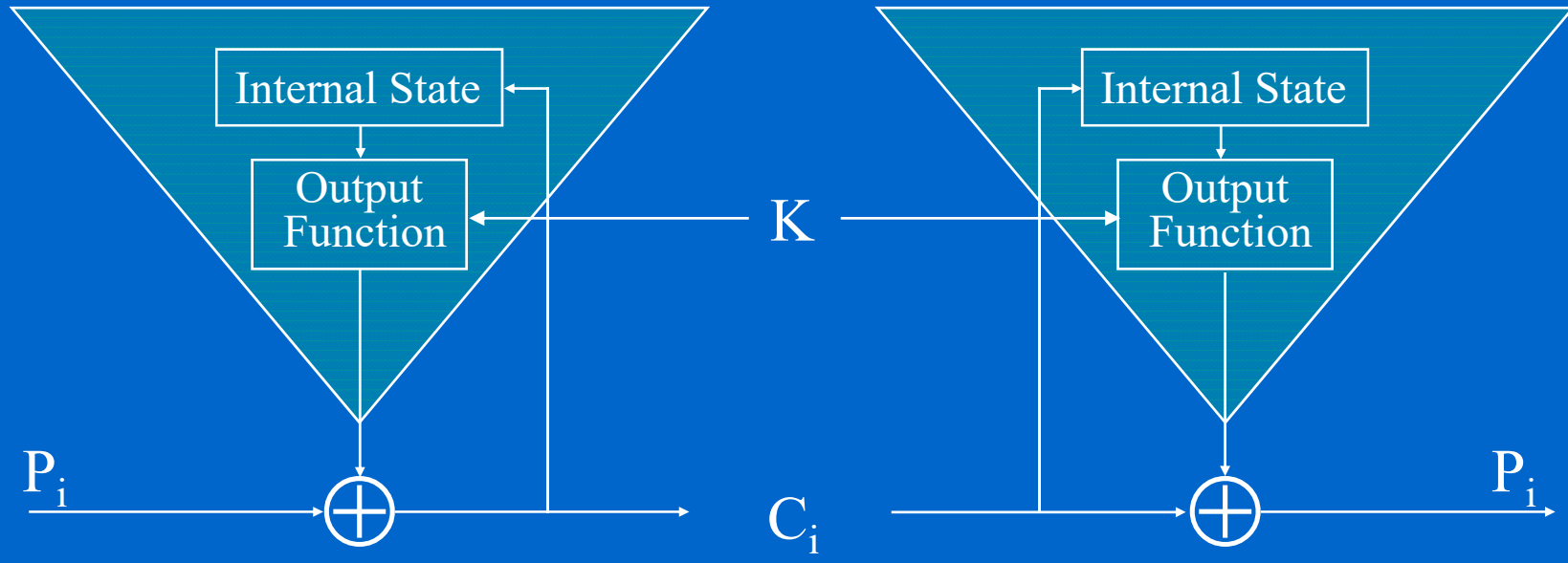
- Security depends entirely on the insides of the keystream generator.
- Key stream generator should be deterministic such that it can be flawlessly reproduced at decryption time.

Key Stream Generator



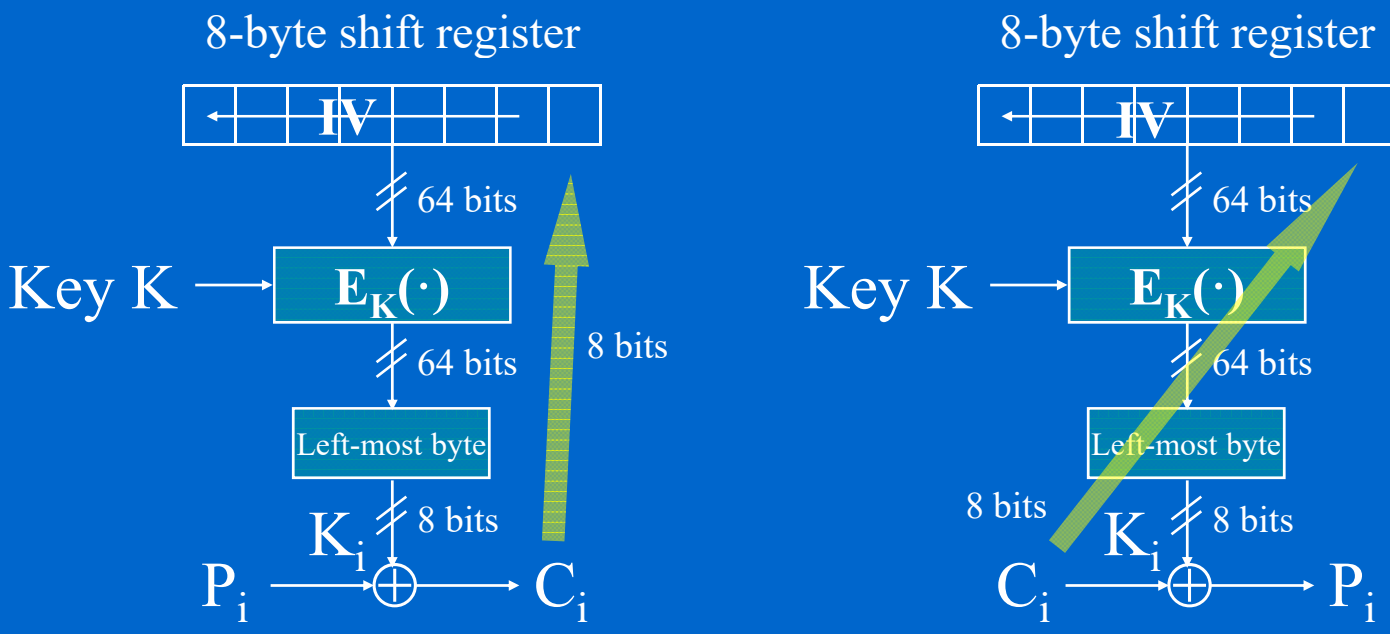
- Two keystream generators, with the same key and the same internal state, will produce the same keystream.

Self-Synchronizing Stream Cipher



- Each keystream bit is a function of a fixed number of previous *n* ciphertext bits.
- The decryption keystream generator will automatically synchronize with the encryption keystream generator after receiving *n* ciphertext bits.

Cipher-Feedback (CFB) Mode

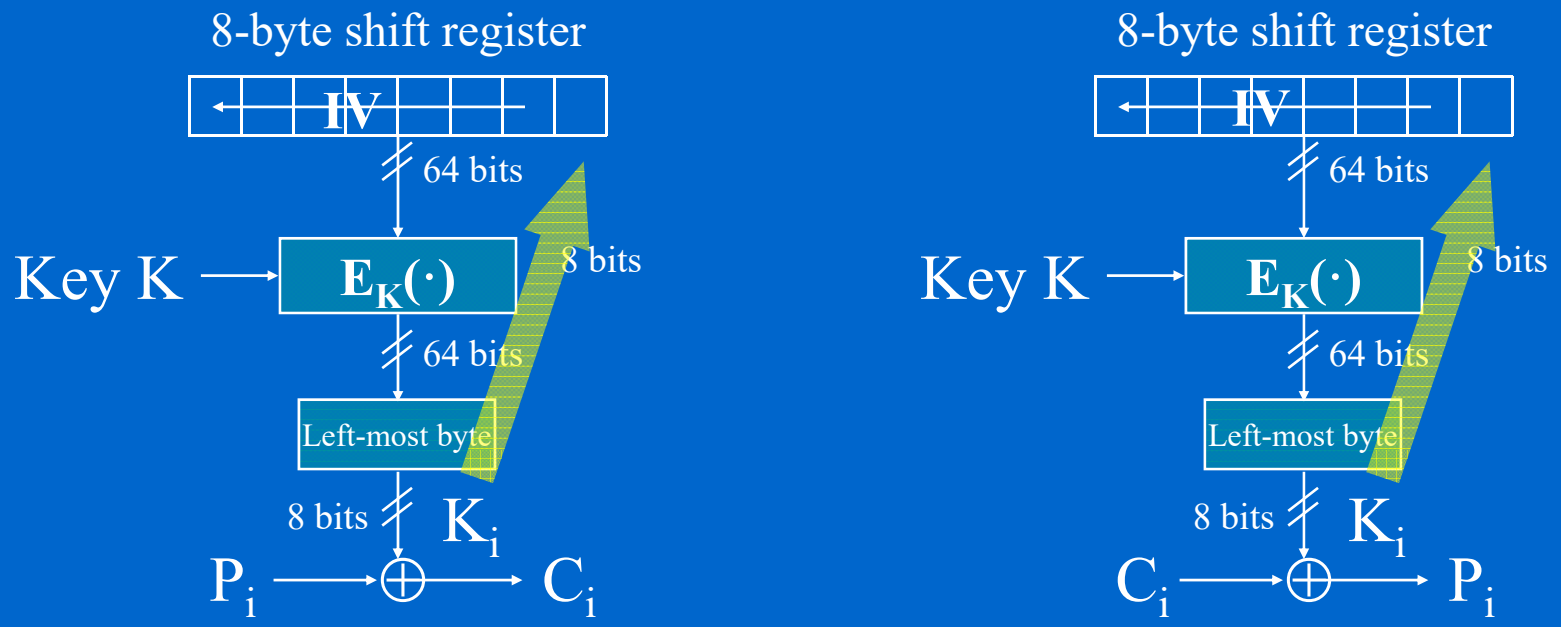


- One bit error in the ciphertext will incorrectly produce *n* keystream errors.
- Playback attack: Bob will resynchronize automatically.
- If IV and plaintext are the same, the key stream and the ciphertext stream would be the same

Synchronous Stream Cipher

- The keystream is generated independent of the message stream. (Key Auto-Key (KAK) system)
- Do not propagate transmission errors.
- Deterministic keystream generator: it must generate the same output keystream on both sides.
- The keystream sequence will eventually repeat. Except one-time pads, all keystream generators are periodic.

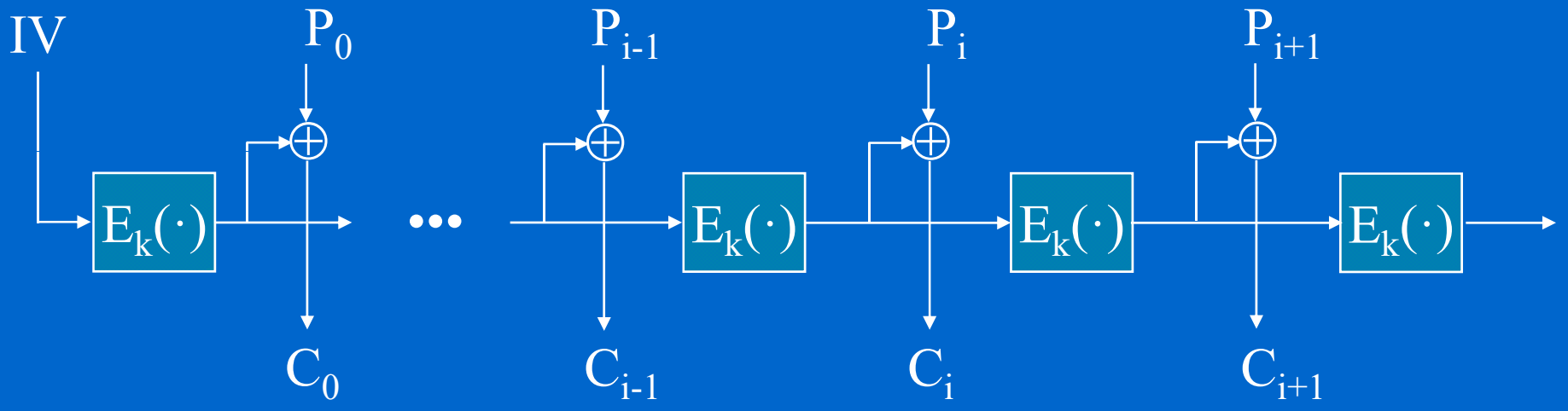
Output Feedback (OFB) Mode



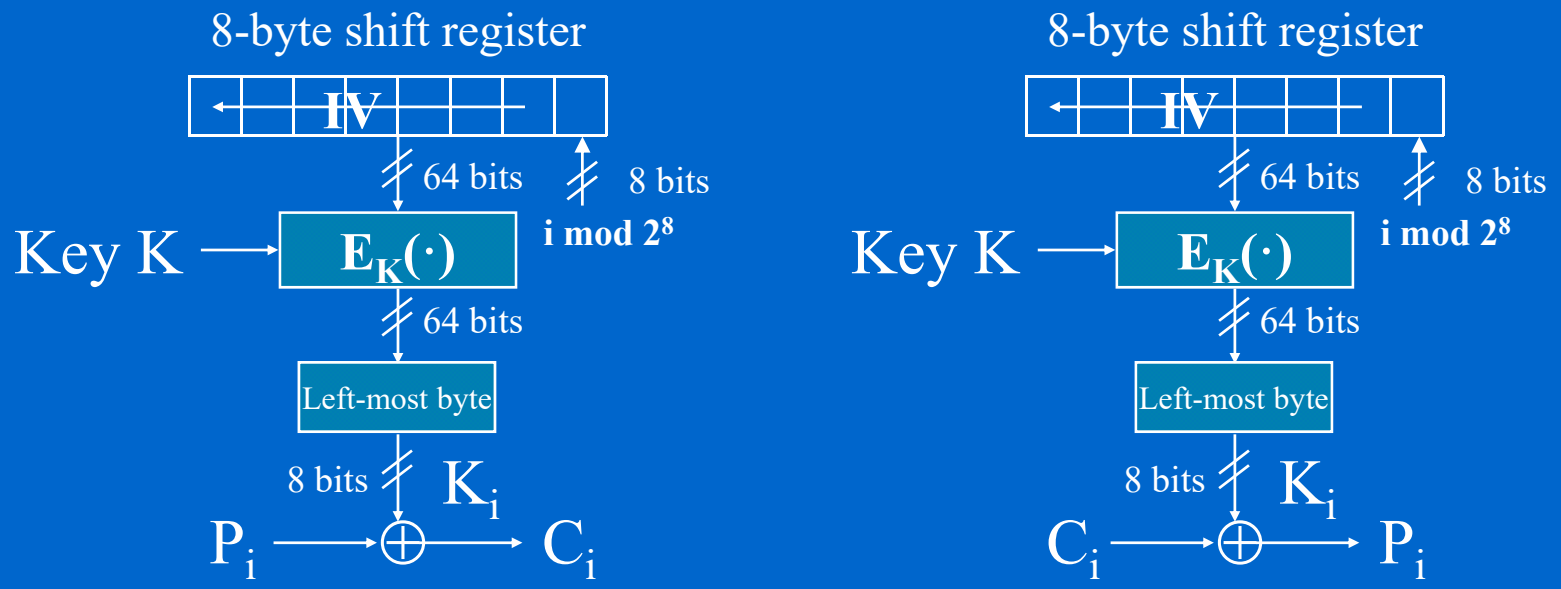
- running a block cipher as a synchronous stream cipher
- feedback mechanism is independent of both the plaintext and the ciphertext streams
- most key generation works can be done offline
- IV should be unique but can be public. The same key stream should not be used twice, e.g. $c_1 = p_1 \oplus k$, $c_2 = p_2 \oplus k$ then
  $p_2 = c_2 \oplus (c_1 \oplus p_1)$

Output Feedback (OFB) Mode

- A single bit error in the ciphertext causes a single-bit error in the recovered plaintext.
- A loss of synchronization is fatal. A mechanism to detect "synchronization losses" is required.
- OFB should be used only when the feedback size is the same as the block size for security reasons. (to make the average key cycle as long as possible, 2^{64}-1. Ex. assume 1 bit feedback and IV = 1010…10, it is very likely that the period of K_i is 2 only)



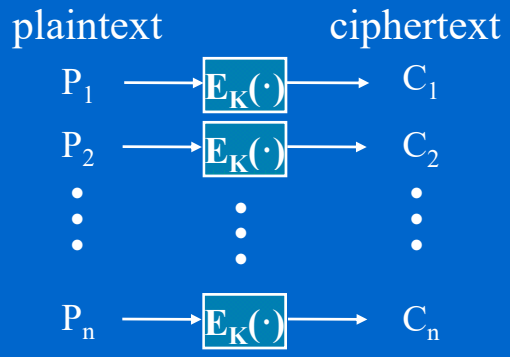
Counter Mode



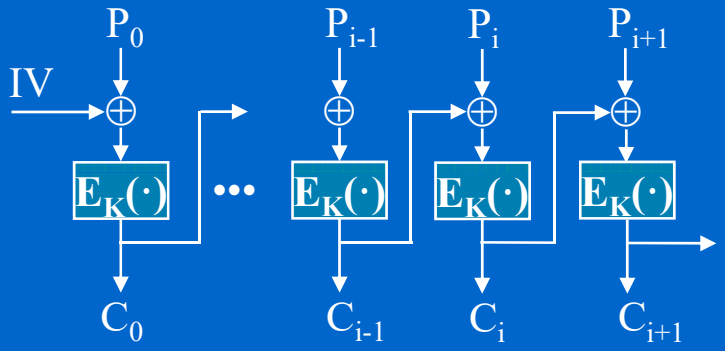
- Just using sequence number as the input to the shift register (or you can use any random-sequence generators, whether cryptographically secure or not)

Comparisons on Mode of Operation

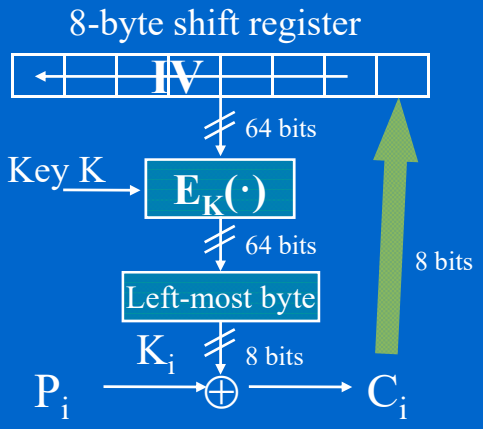
ECB



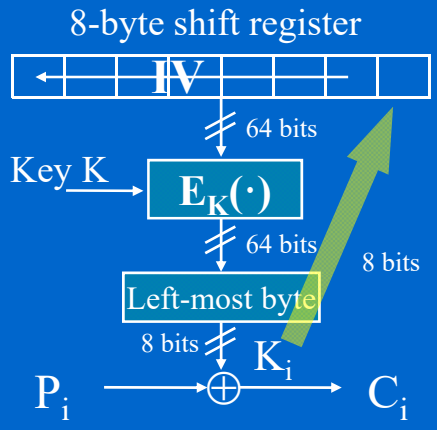
CBC



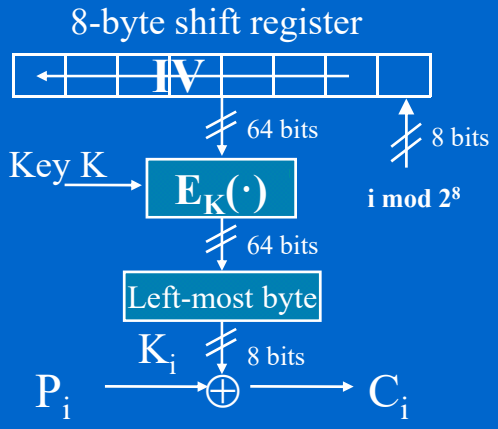
CFB



OFB



Counter



Comparisons - ECB

Bruce Schneier, "Applied Cryptography"

ECB:

Security:

- Plaintext patterns are not concealed. (the same plaintext maps to the same ciphertext.)
- Input to the block cipher is not randomized; it is the same as the plaintext.
+ More than one message can be encrypted with the same key.
- Plaintext is easy to manipulate; blocks can be removed, repeated or interchanged.

Efficiency:

+ Speed is the same as the block cipher.
- Ciphertext is up to one block longer than the plaintext, due to padding.
- No preprocessing is possible.
- Processing is parallelizable.

Fault-tolerance:

- A ciphertext error affects one full block of plaintext.
- Synchronization error is unrecoverable.

Comparisons - CBC

CBC:

Security:

+ Plaintext patterns are concealed by XORing with previous ciphertext block.
+ Input to the block cipher is randomized by XORing with the previous ciphertext block.
+ More than one message can be encrypted with the same key.

+/- Plaintext is somewhat difficult to manipulate.

- One bit error in causes an error block and one error bit in the following block.
- Removal of ciphertext blocks causes errors of corresponding message blocks.
- Insertion of m ciphertext blocks causes m+1 errors in the plaintext blocks.
- Repetition is kind of insertion.
- Swapping of 2 ciphertext blocks causes 4 blocks of errors.

Comparisons - CBC

Efficiency:

+ Speed is the same as the underlying block cipher.
- Ciphertext is up to one block longer than the plaintext, not counting the IV.
- No preprocessing is possible.
+/- Encryption is not parallelizable, decryption is parallelizable and has a random-access property.

Fault-tolerance:

- A ciphertext error affects one full block of plaintext and the corresponding bit in the next block
- Synchronization error is unrecoverable

Comparisons - CFB

CFB:

Security:

+ Plaintext patterns are concealed since the key depends on previous ciphertext stream.
+ Input to the block cipher is previous ciphertext stream.
+ More than one message can be encrypted with the same key. If a different IV is used, ciphertext stream will not be the same.
+/- Plaintext is somewhat difficult to manipulate, blocks can be removed from the beginning and end of the message, bits of the first block can be changed, and repetition allows some controlled changes.

Efficiency:

+ Speed is the same as the block cipher.
- Ciphertext is the same size as the plaintext, not counting the IV.

Comparisons on Mode of Operation

+/- Encryption is not parallelizable, decryption is parallelizable and has a random-access property.

- Some preprocessing is possible before a block is seen; the previous ciphertext block can be encrypted.

+/- Encryption is not parallelizable; decryption is parallelizable and has a random-access property.

Fault-tolerance:

- A ciphertext error affects the corresponding bit of plaintext and the next full block.

+ Synchronization errors of full block sizes are recoverable. 1-bit CFB can recover from the addition or loss of single bits.

Comparisons – OFB/Counter

OFB/Counter:

Security:

+ Plaintext patterns are concealed. Different keys might be used for the same plaintext.
+ Input to the block cipher is previous keystream.
+ More than one message can be encrypted with the same key, provided that a different IV is used.
- Plaintext is very easy to manipulate; any change in ciphertext directly affects the plaintext.

Efficiency:

+ Speed is the same as the clock cipher.
- Ciphertext is the same size as the plaintext, not counting the IV.

Comparisons – OFB/Counter

+ Processing is possible before the message is seen
-/+ OFB processing is not parallelizable; counter processing is parallelizable.

Fault-tolerance:
+ A ciphertext error affects only the corresponding bit of plaintext.
- Synchronization error is unrecoverable.

Breaking DES

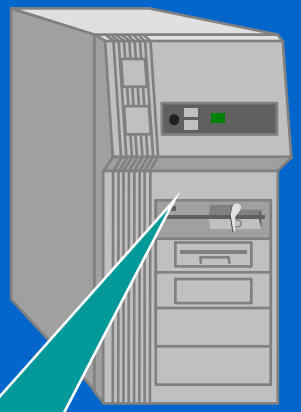
- Brute-force attacks:
  - distributive computation
    - Rocke Verser: more than 1000 computers on the Internet, search over 1/4 of the key space for 5 months to find the 1997 RSA Data Security's DES Challenge for a prize of US$10000
  - custom architecture
    - Electronic Frontier Foundation (EFF)'s 'DES cracker': 39 days were used to search 85% of the key space to find the key for RSA Data Security's DES Challenge II.
    - 1 PC, software, 24 search units/chip, 64 chips/board, 12 boards/chassis, 2 chassis (36864 units in total)

Enhanced DES

- Double DES
  - although 112 bits key were used, the security level is the same as a 57 bit scheme
  - meet-in-the-middle attack
- Triple DES
  - three key system: $E_{K_1}(E_{K_2}(E_{K_3}(m)))$
  - two key system: $E_{K_1}(D_{K_2}(E_{K_1}(m)))$ compatibility
- DESX: $K_3 \oplus (E_{K_2}(K_1 \oplus m))$ by Rivest
- s^nDES
  - redesign S-Boxes such that linear approximations are minimized

Unix Password Security

- Direct password authentication model



password plaintext

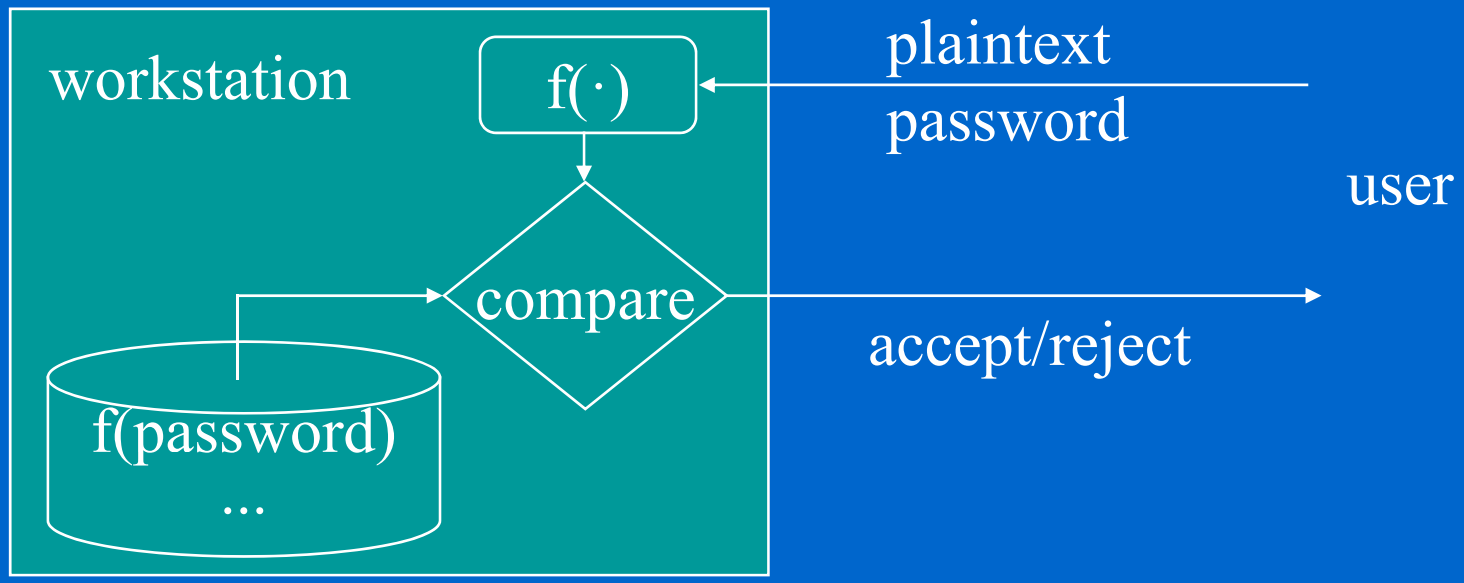
accept/reject

Compare with stored plaintext password

Eve might sneak into the system and steal passwords from the system file.

counter measure: encrypt the password before storing them

Unix Password Security



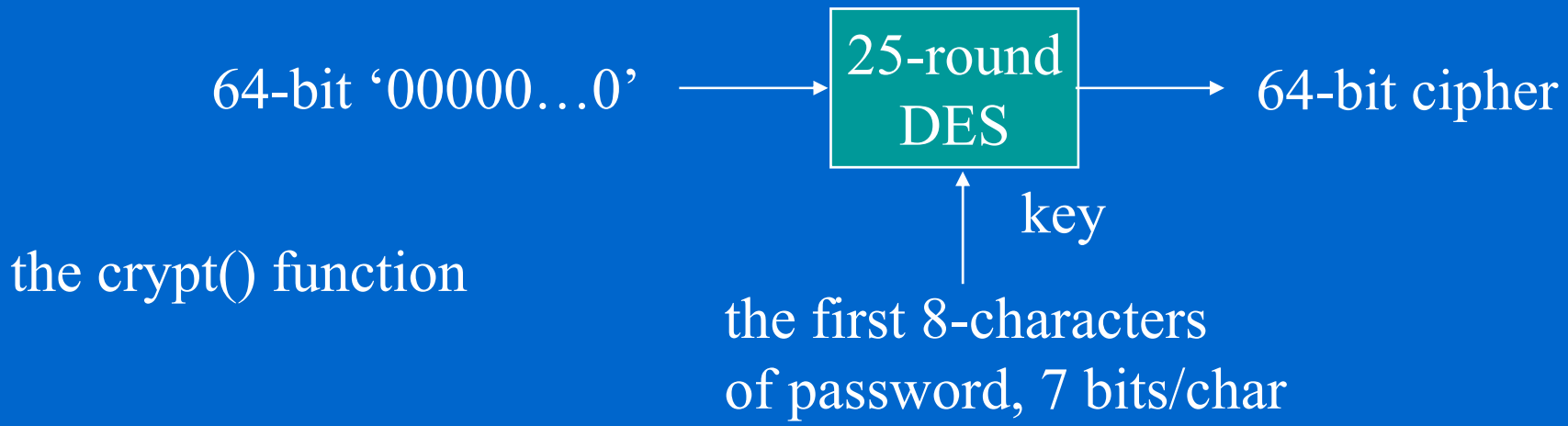
- f(·) is a sort of one-way function (not necessary a permutation); given y = f(x), it's hard to solve x
- To pass the identification check, a user need to key in the plaintext password. Although Eve might have access to the system password file, she still does not know the plaintext password, unless she can invert f(·)

Unix Password Security

- Two types of f(·) function:
  - MD5 hash function:

collision resistant

  - modified DES:



the crypt() function

Unix Password Security

- Dictionary Attacks:
  - people tend to choose meaningful words or their modification as their password
  - greatly reduce the possible set of passwords
    $95^8 \cong 6.6 \times 10^{15} \Rightarrow 80000$
  - although it's hard to invert $f(\cdot)$; now that we have the possible set of passwords, we can try every possible f(password) explicitly, and match with the user entries in the system file; even use H/W DES cracker
  - counter measure: salt

Unix Password Security

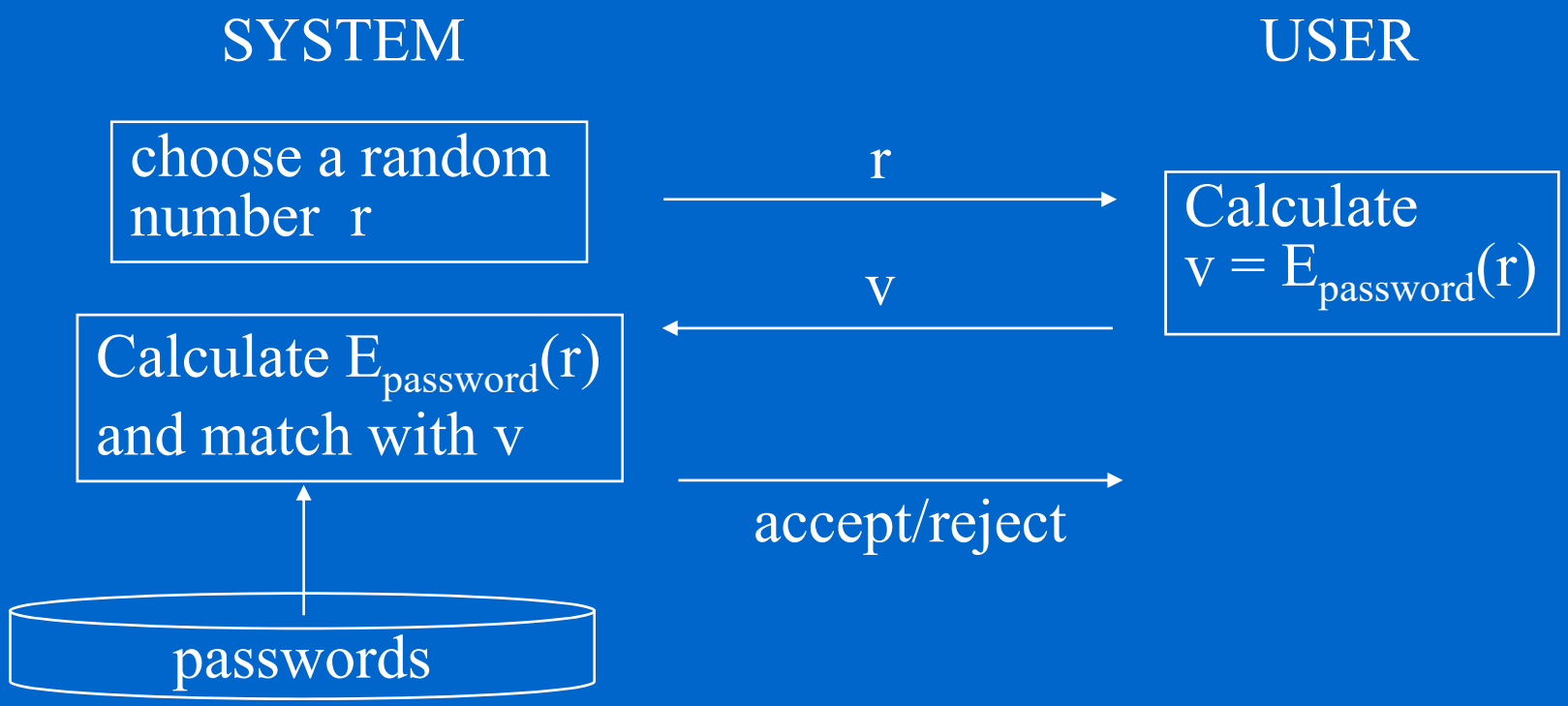
- Salt:
  - additional 12 bits (2 characters, each from 64 candidates)
  - together with the 8-character password determines the ciphertext stored in the system file
  - two users using the same password would not have the same ciphertext entry in the system file
  - make a general dictionary attack to all user harder, although it is the same to attack an individual's password because the salt value for a particular user is publicly known

Unix Password Security

- Usage of the 12-bit salt:
  - E(R) in the DES round function is a 32→48 bit mapping
  - swap bit 1 and bit 25 if bit 1 of the salt is 1, else no swap
  - swap bit 2 and bit 26 if bit 2 of the salt is 1, else no swap
  - …
  - custom DES algorithm avoids the attack of a hardware ‘DES cracker’

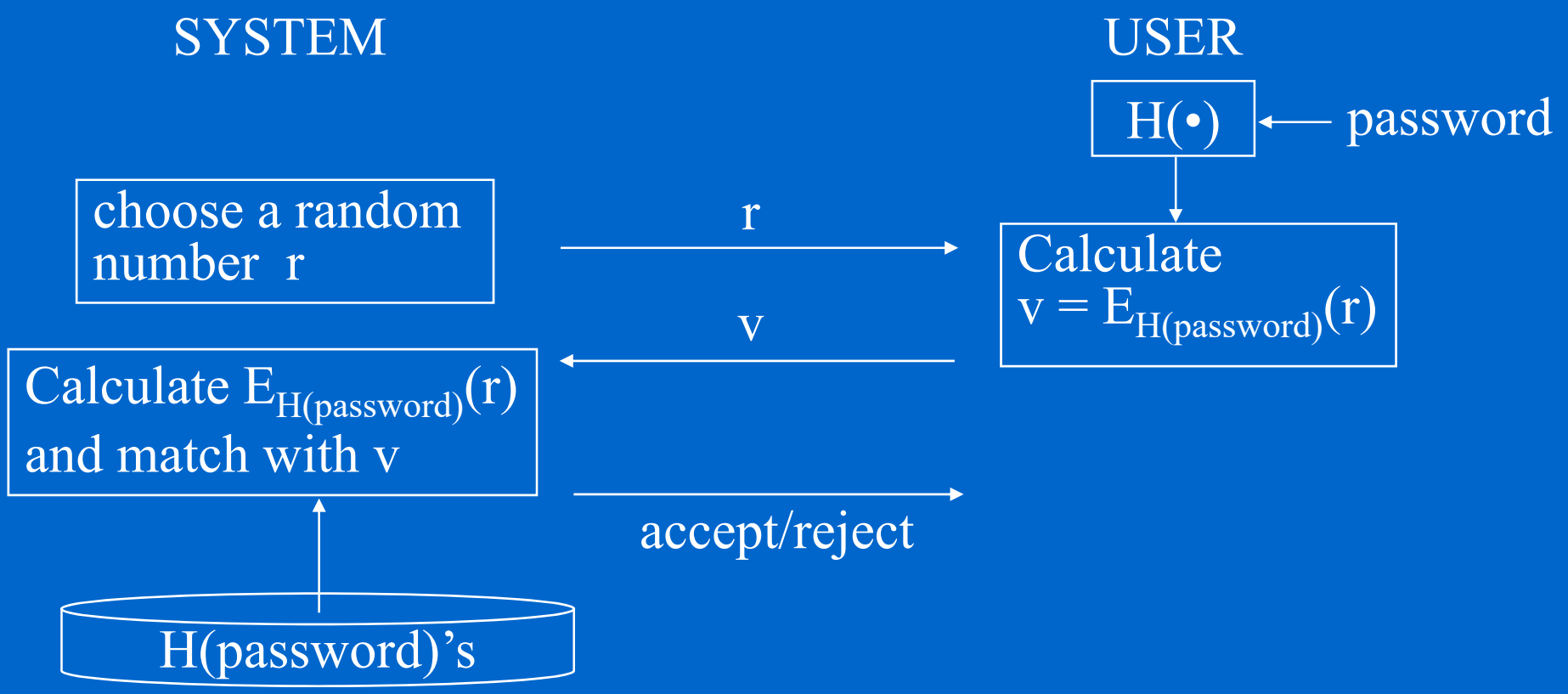
Challenge-Response Password Auth.

- Challenge-Response Protocol
  - to avoid transmitting the plaintext password, to avoid the replay attack



Challenge-Response Password Auth.

- How do we avoid the attack on the public password file? Is there a method for the system to compare the response without storing the passwords for all the users?



Challenge-Response Password Auth.

- A cryptographic collision-resistant hash function H(•) (ex. MD5, SHA1…) can be used instead of E_k(•).