Block Ciphers

Codebook Cipher

Original codebook

Word	Code
The	001
good	002
staff	003
dog	004
cat	005

Additive codebook

Code
058
021
001
092
222
111
087
022
044

3

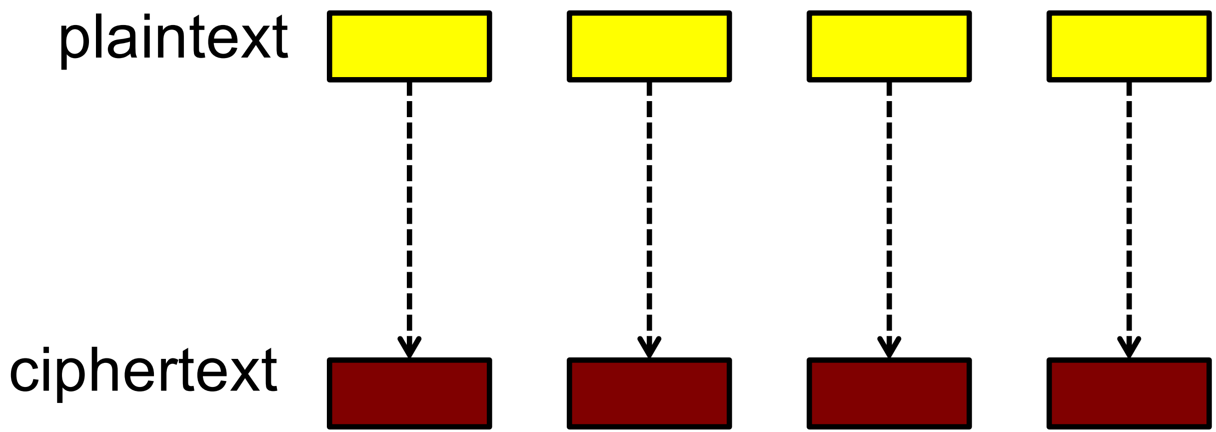
Plaintext: good dog → 002 004

Random additive: 3

Ciphertext: 094 226

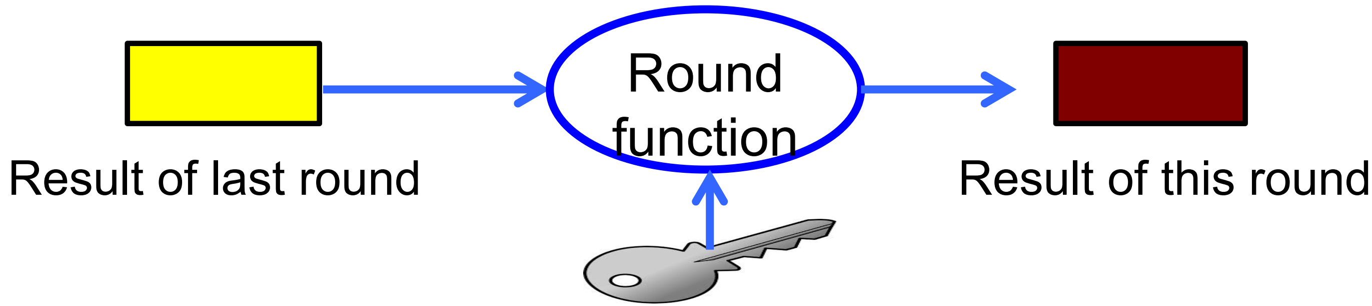
Block Cipher

- Plaintext and ciphertext consist of fixed-sized blocks



Iterated Block Cipher

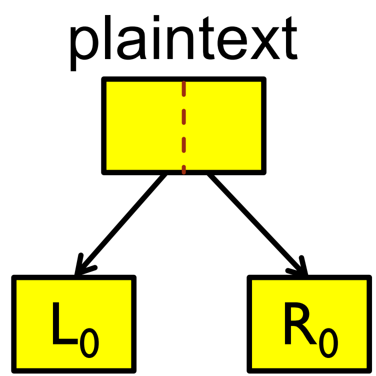
- Ciphertext obtained from plaintext by iterating a **round function**
- Input to round function consists of ***key*** and ***output*** of previous round



- Usually implemented in software

Feistel Cipher

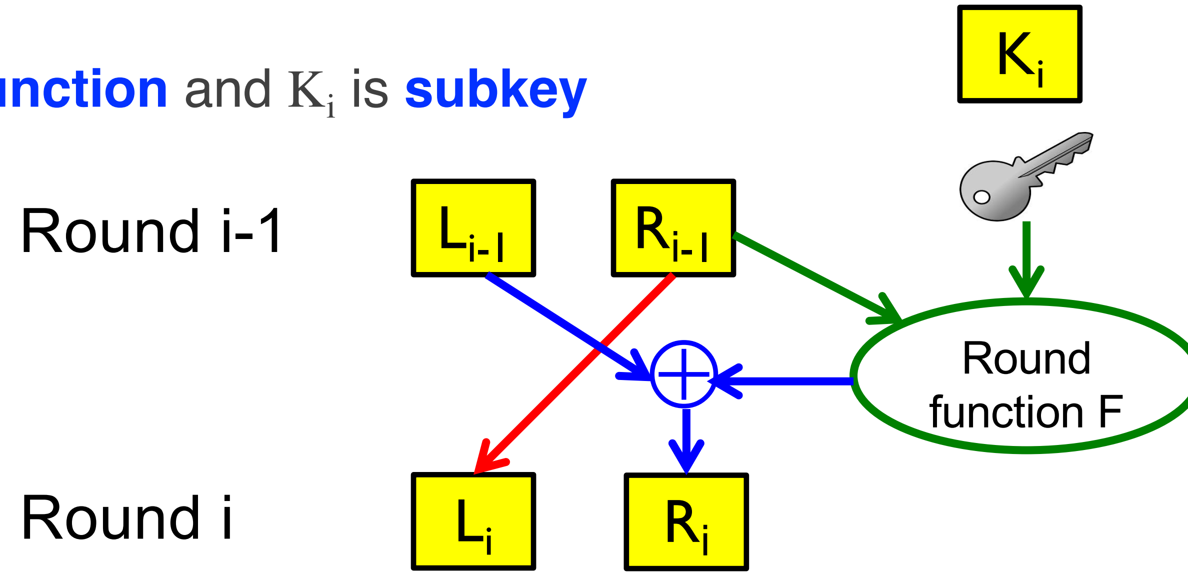
- Named after **Horst Feistel**, who did this work at IBM
- **Feistel cipher** is a **principle** of block cipher, not a specific block cipher
- Split plaintext block into left and right halves: $P = (L_0, R_0)$



Feistel Cipher: Encryption

- For each round $i = 1, 2, \ldots, n$, compute

  $L_i = R_{i-1}$
  $R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$
  where F is **round function** and K_i is **subkey**



- Ciphertext: $C = (L_n, R_n)$

What is $\mathbf{L_{i-1}}$ and $\mathbf{R_{i-1}}$ given $\mathbf{L_i}$, $\mathbf{R_i}$, and $\mathbf{K_i}$?

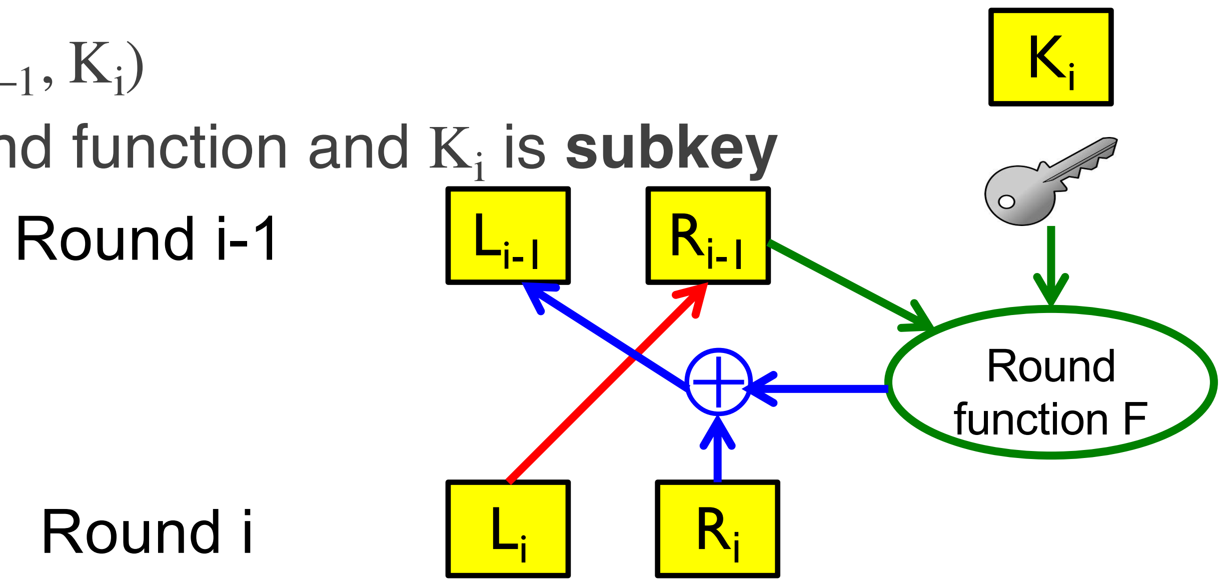
Feistel Cipher: Decryption

- Start with ciphertext $C = (L_n, R_n)$
- For each round $i = n, n-1, \ldots, 1$, compute

  $R_{i-1} = L_i$

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

  where F is round function and K_i is **subkey**



- Plaintext: $P = (L_0, R_0)$

Key schedule

- A key schedule is the algorithm to generate the subkey in each round from the original key

For each round $i = 1, 2, \ldots, n$, compute

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

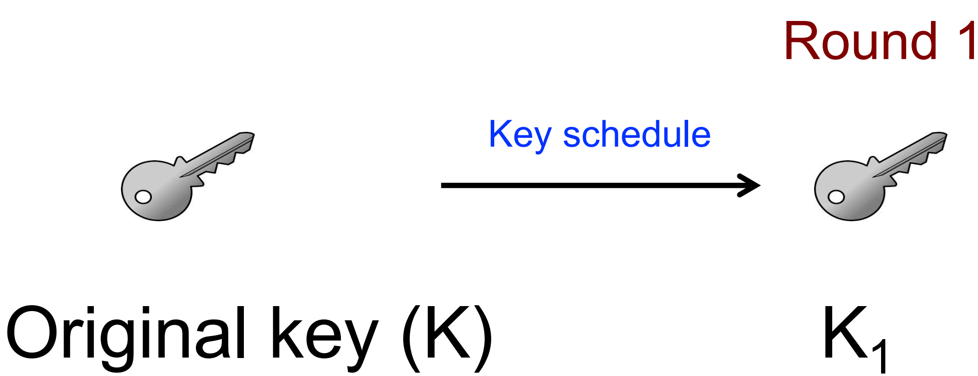
Original key (K)

Key schedule

- A key schedule is the algorithm to generate the subkey in each round from the original key

For each round $\mathrm{i} = 1, 2, \ldots, \mathrm{n}$, compute

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



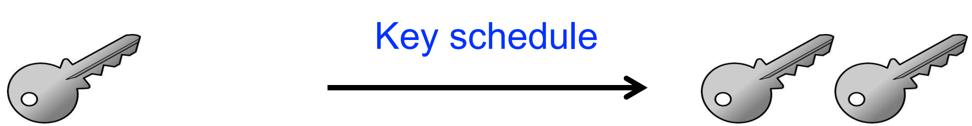
Key schedule

- A key schedule is the algorithm to generate the subkey in each round from the original key

For each round $i = 1, 2, \ldots, n$, compute

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

Round 2



Original key (K) K_1 K_2

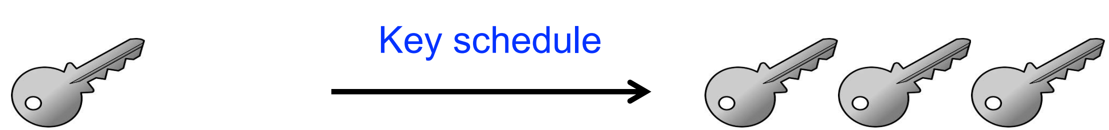
Key schedule

- A key schedule is the algorithm to generate the subkey in each round from the original key

For each round $i = 1, 2, \ldots, n$, compute

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

Round 3



More about Feistel cipher

- Formula "works" for any function F, but only secure for certain functions F

- The encryption and decryption operations are very similar, requiring only the reversal of the key schedule (how subkey is obtained)

- Hence, the size of the **code** (if implemented in software) or the **circuitry** (if implemented by hardware) is reduced.

Data Encryption Standard (DES)

DES - History

- In the 1970's, realized that there was a commercial need for crypto
- Call for cipher proposals by NBS (National Bureau of Standards, now known as NIST) in the mid 1970's to become a US government standard
- IBM's Lucifer algorithm the only serious contender
- Little Crypto expertise at NBS; asked for help from NSA(National Security Agency)
- NSA agreed to get involved, on the condition that its role wouldn't become public
- Subtle changes to Lucifer algorithm, such as key length reduced from 128 to 56 bits
- Approved as US standard in 1976

DES - History

- There was suspicion that NSA put a backdoor in DES, but 30 years of intense cryptanalysis has revealed no backdoor in DES
- In the 70's, both IBM and NSA knew differential cryptanalysis, but didn't publish it
  - **Differential cryptanalysis**: the study of how differences in the information input can affect the resultant difference at the output
- It was found that DES's **S-boxes** were designed to repel differential cryptanalysis

DES Numerology

- DES is a Feistel cipher with…
  - 64 bit block length
  - 56 bit key length
  - 16 rounds
  - 48 bits of key used each round (**subkey**)
- Each round is simple (for a block cipher)
- Security depends heavily on "**S-boxes**"

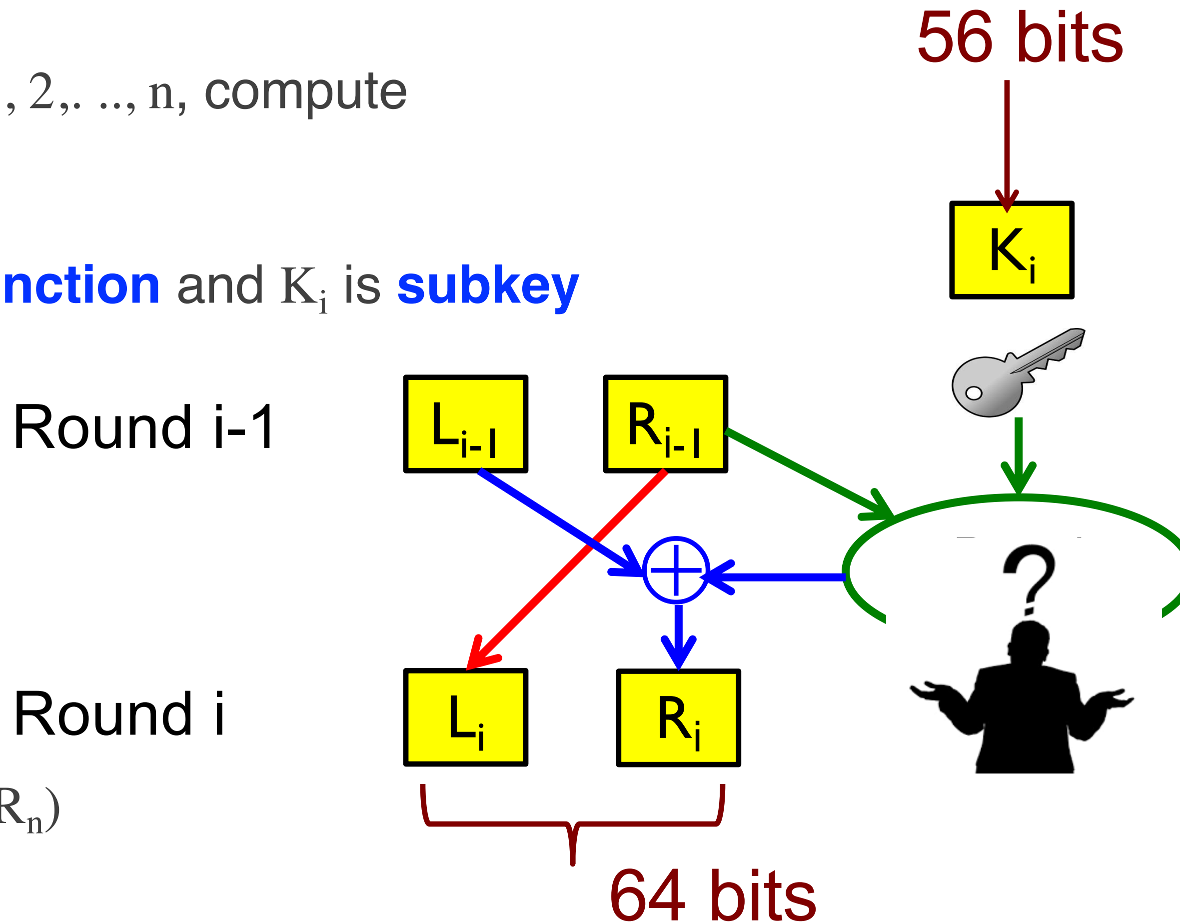
DES is a Feistel Cipher

- For each round $i = 1, 2, \ldots, n$, compute

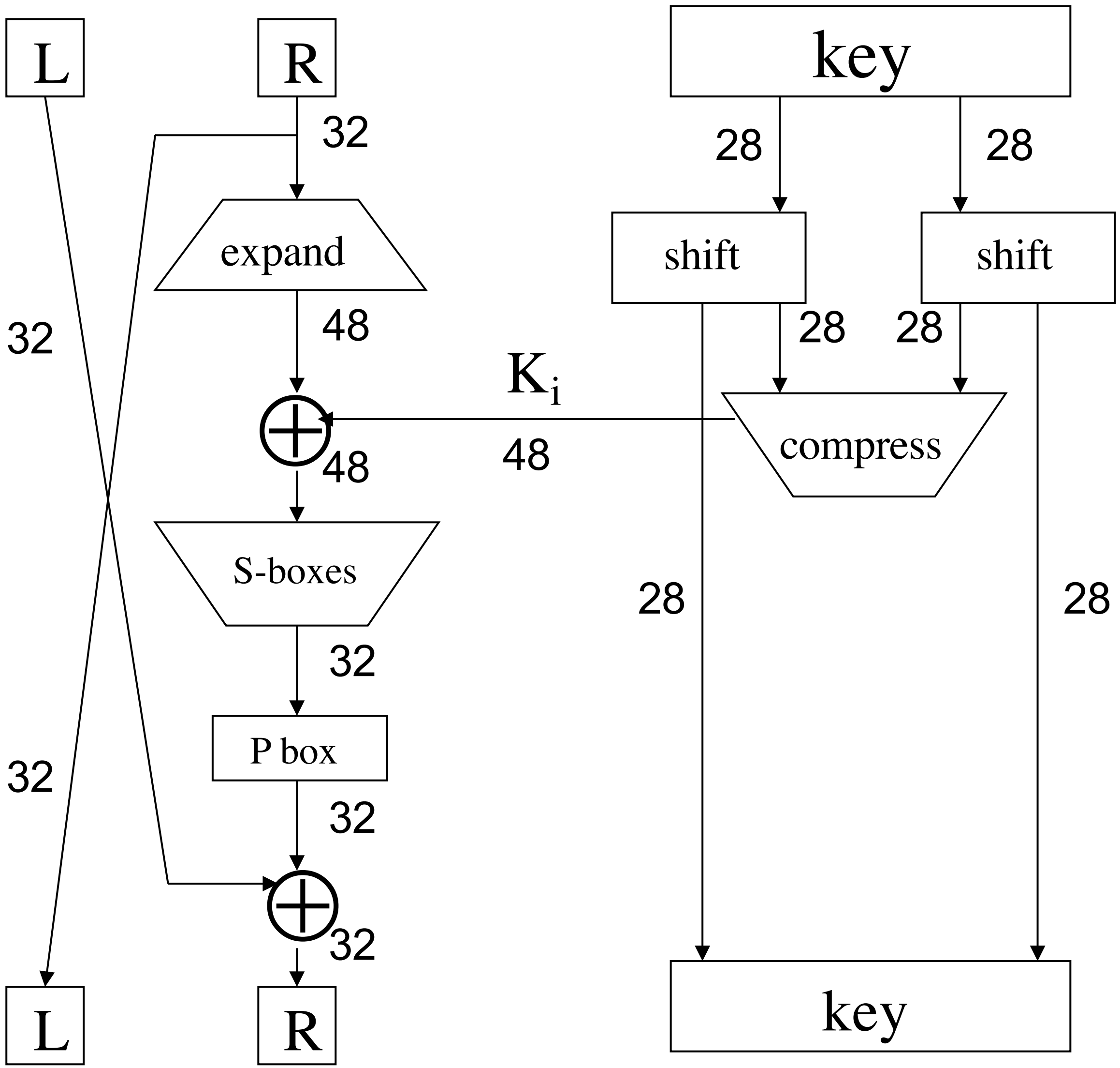
  $L_i = R_{i-1}$

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

  where F is **round function** and K_i is **subkey**

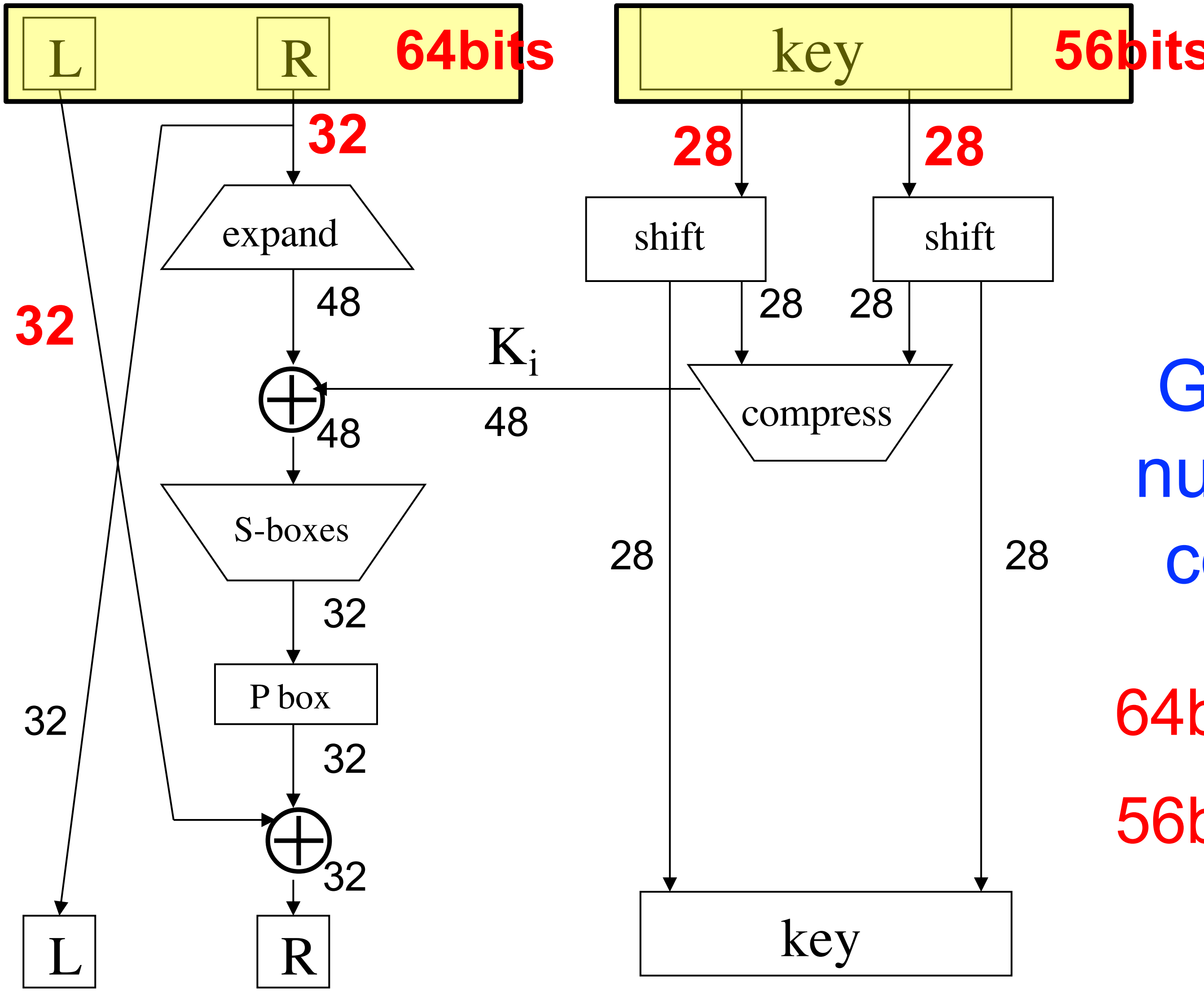


- Ciphertext: $C = (L_n, R_n)$



One Round of DES

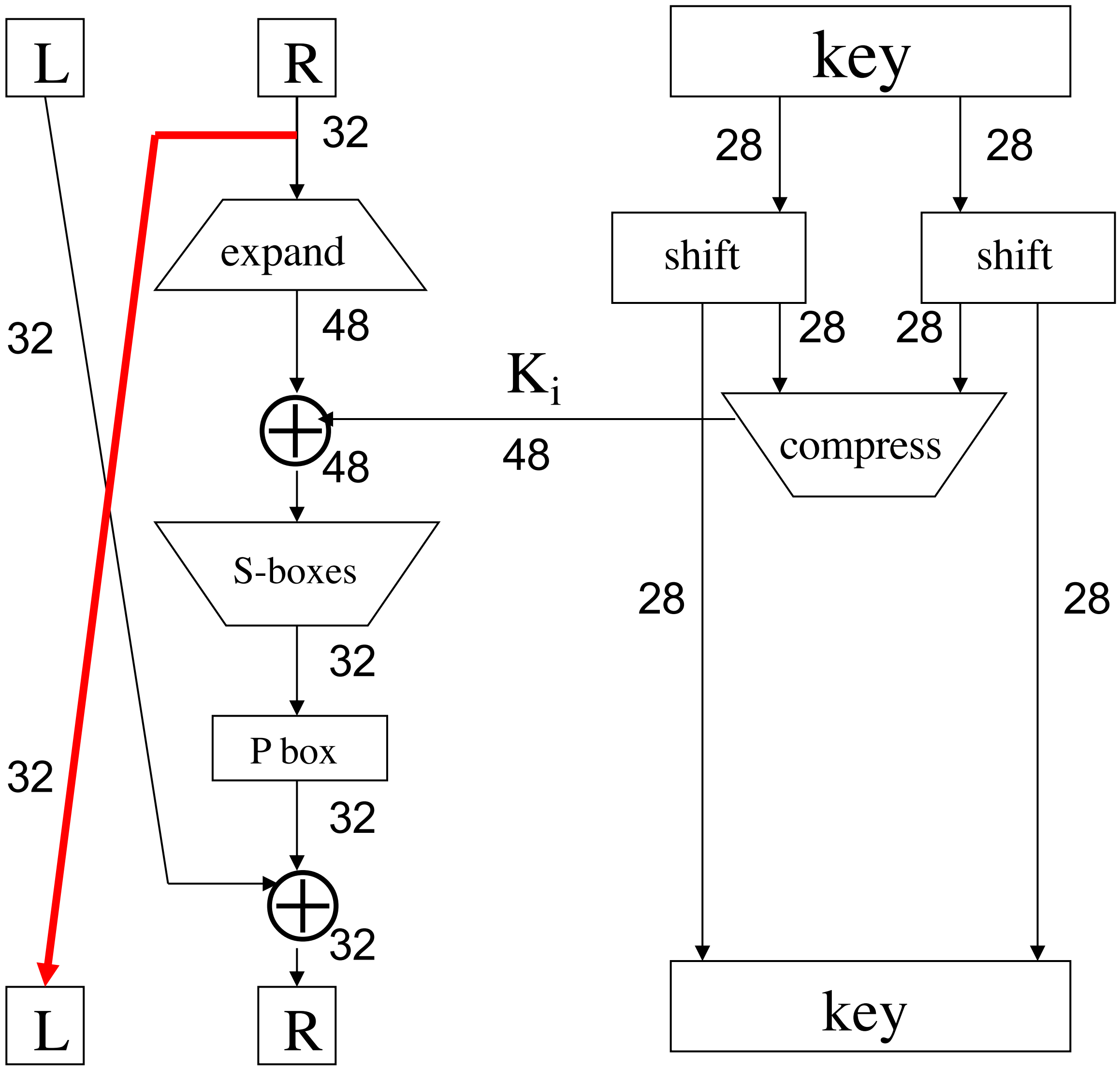
(in total: 16 rounds)



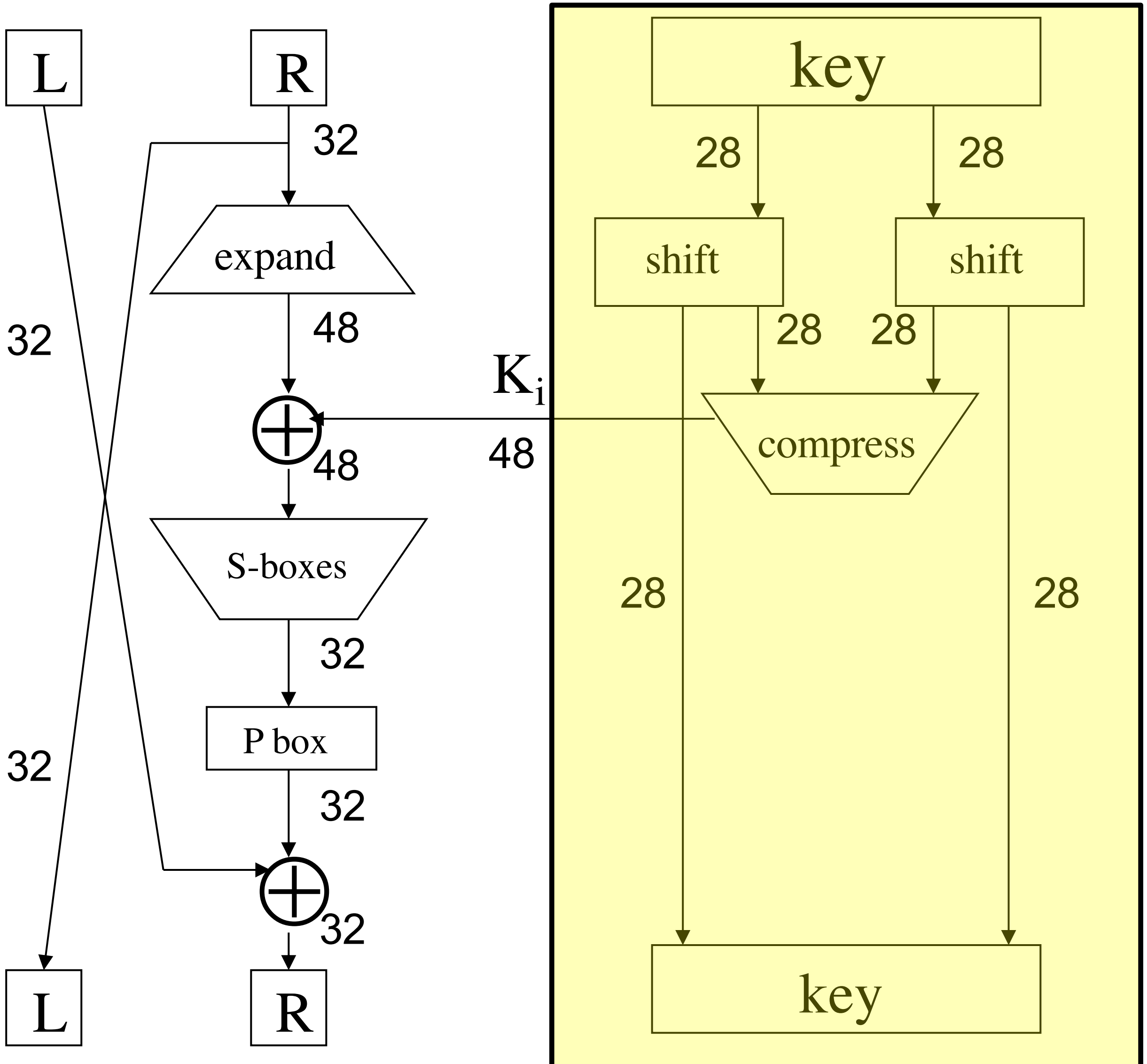
Get the numbers correct

64bit block

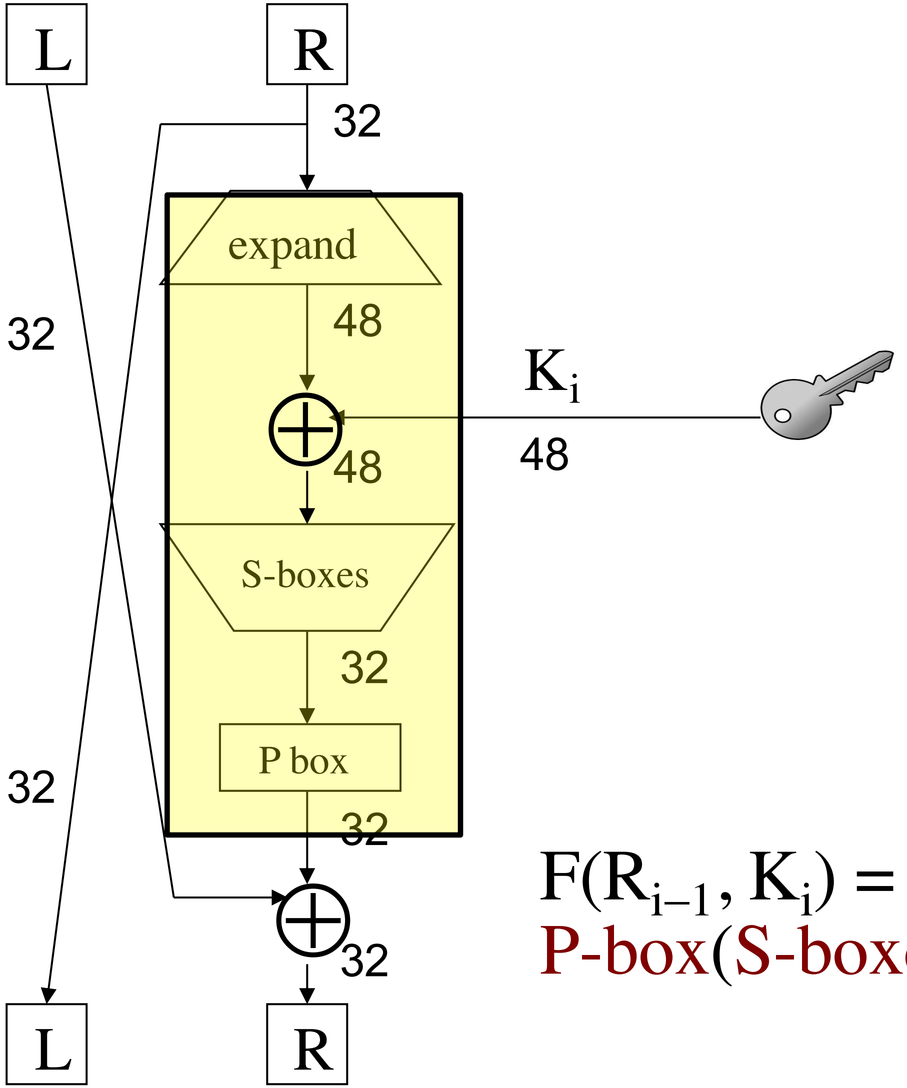
56bit key



Step 1 in Feistel cipher

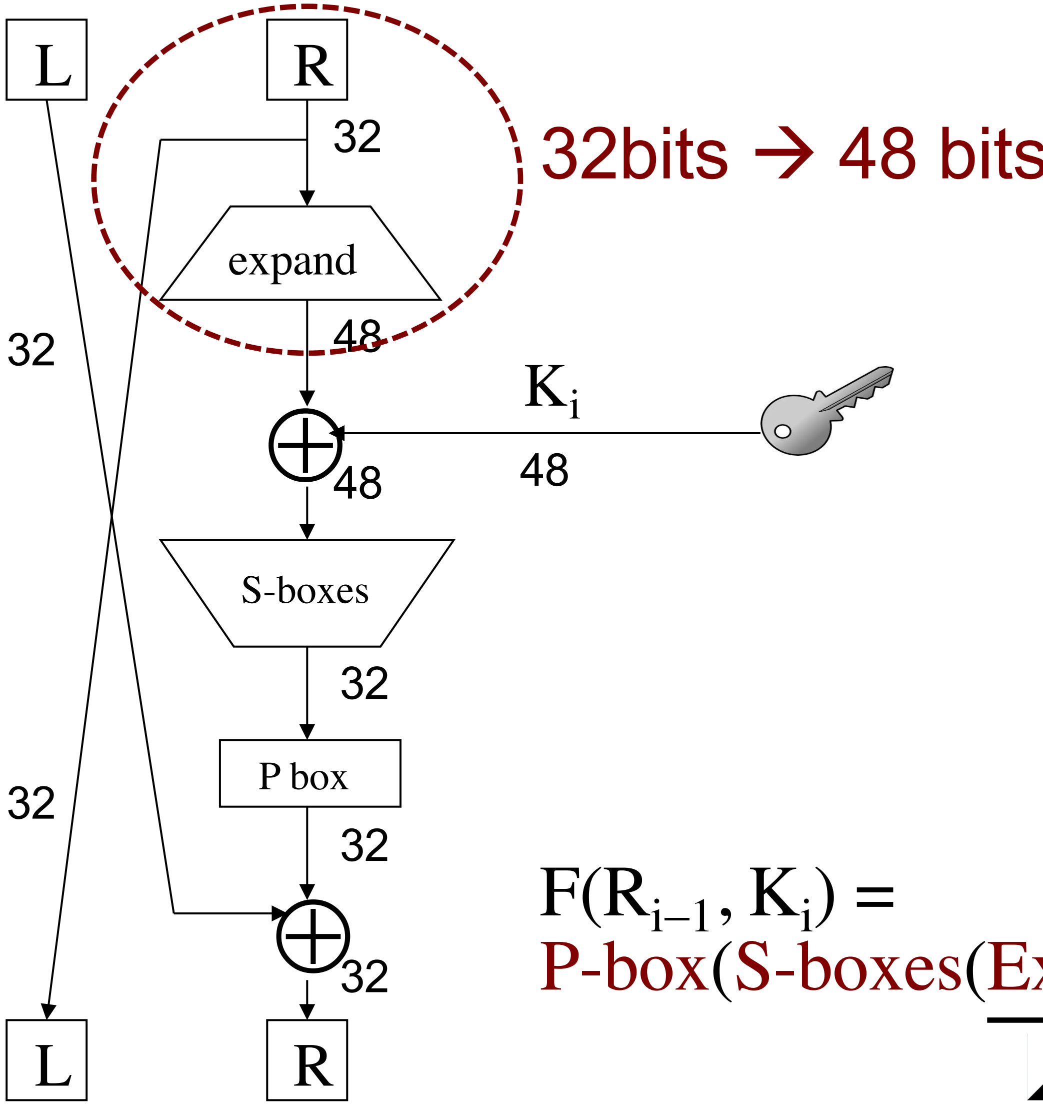

L
R
32
expand
48
32
K_i
48
48
S-boxes
32
P box
32
32
32
32
L
R
key
28
28
shift
shift
28
28
compress
28
28
key

Subkey



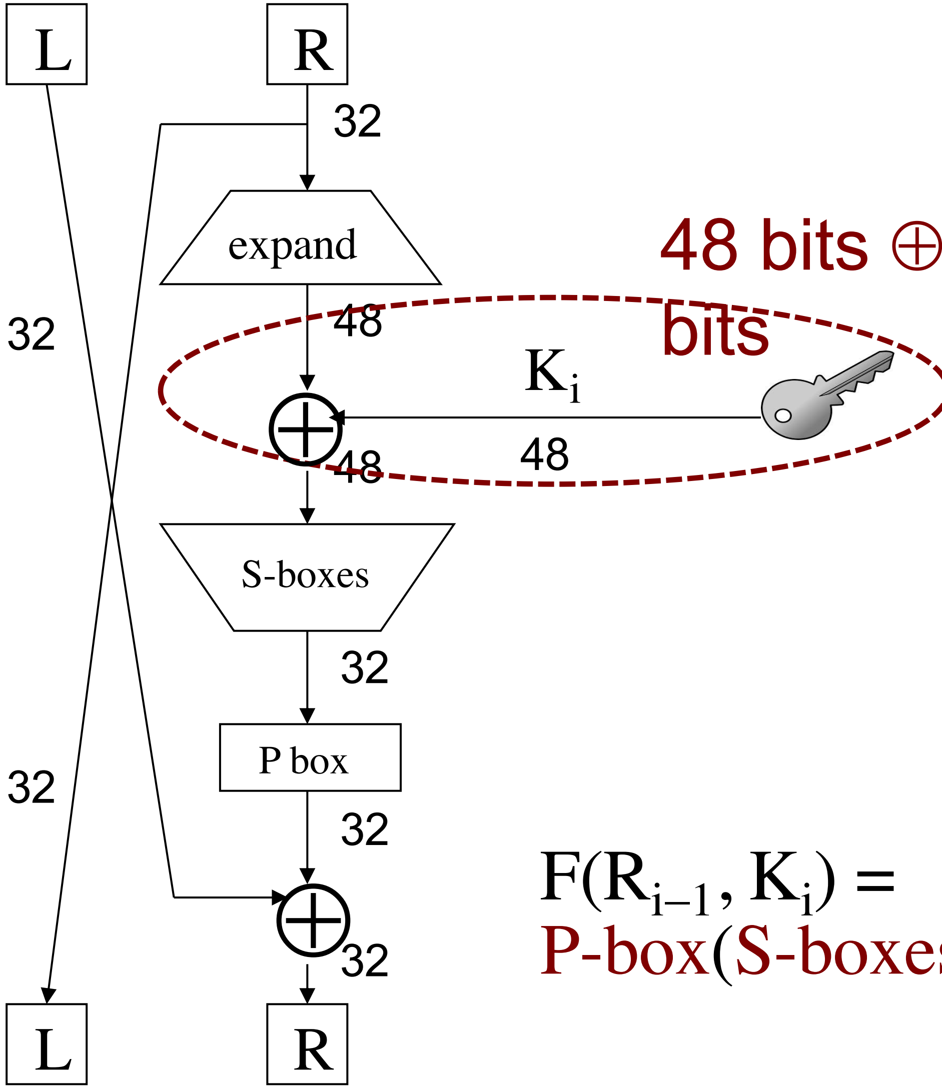
Round function

$$F(R_{i-1}, K_i) = \text{P-box}(\text{S-boxes}(\text{Expand}(R_{i-1}) \oplus K_i))$$



Round function

$$F(R_{i-1}, K_i) = \text{P-box}(\text{S-boxes}(\underline{\text{Expand}(R_{i-1})} \oplus K_i))$$



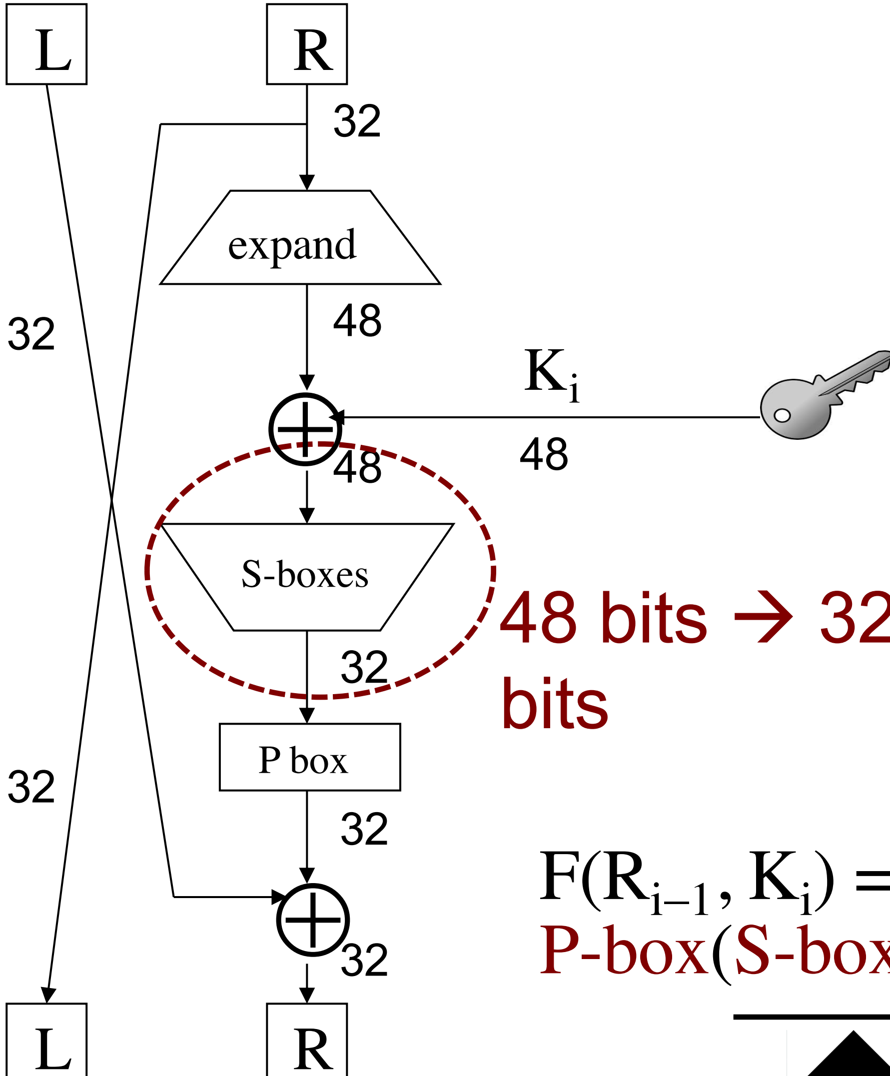
48 bits ⊕ 48 bits

Round function

$F(R_{i-1}, K_i) =$

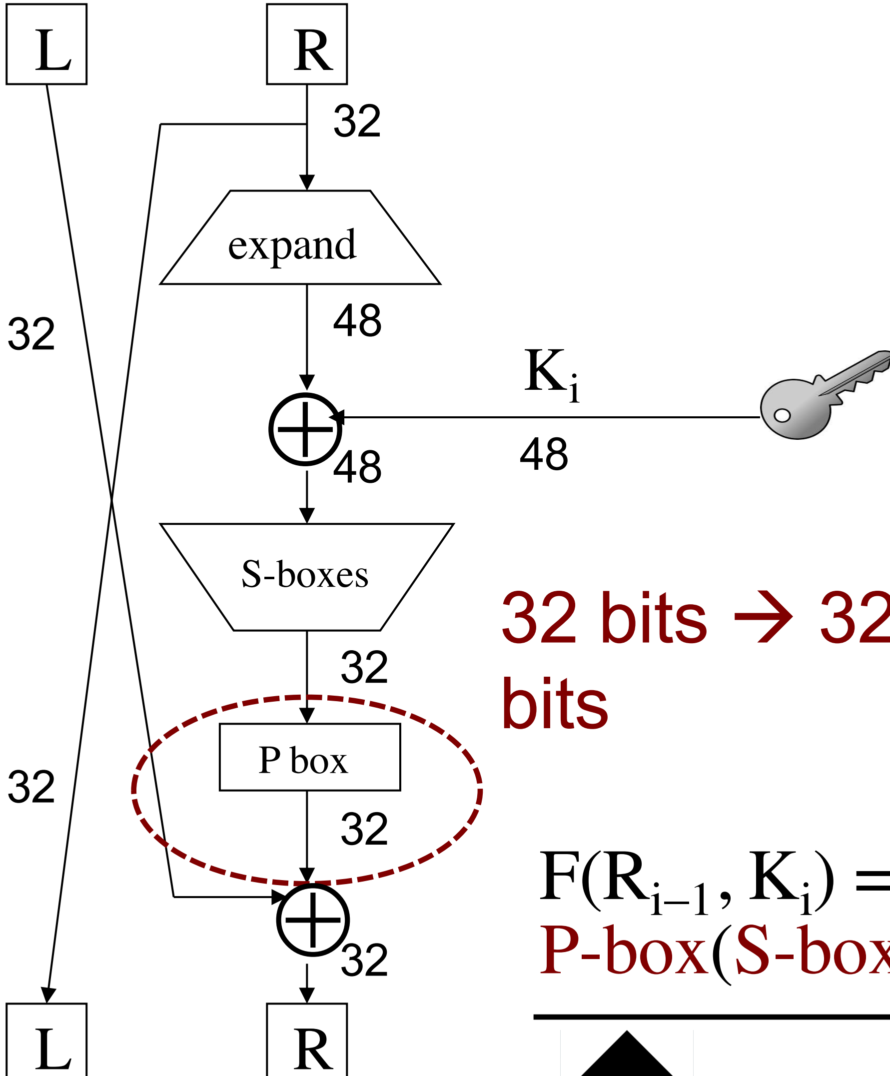
P-box(S-boxes(Expand(R_{i-1})⊕K_i))

BINGHAMTON UNIVERSITY
STATE UNIVERSITY OF NEW YORK



Round function

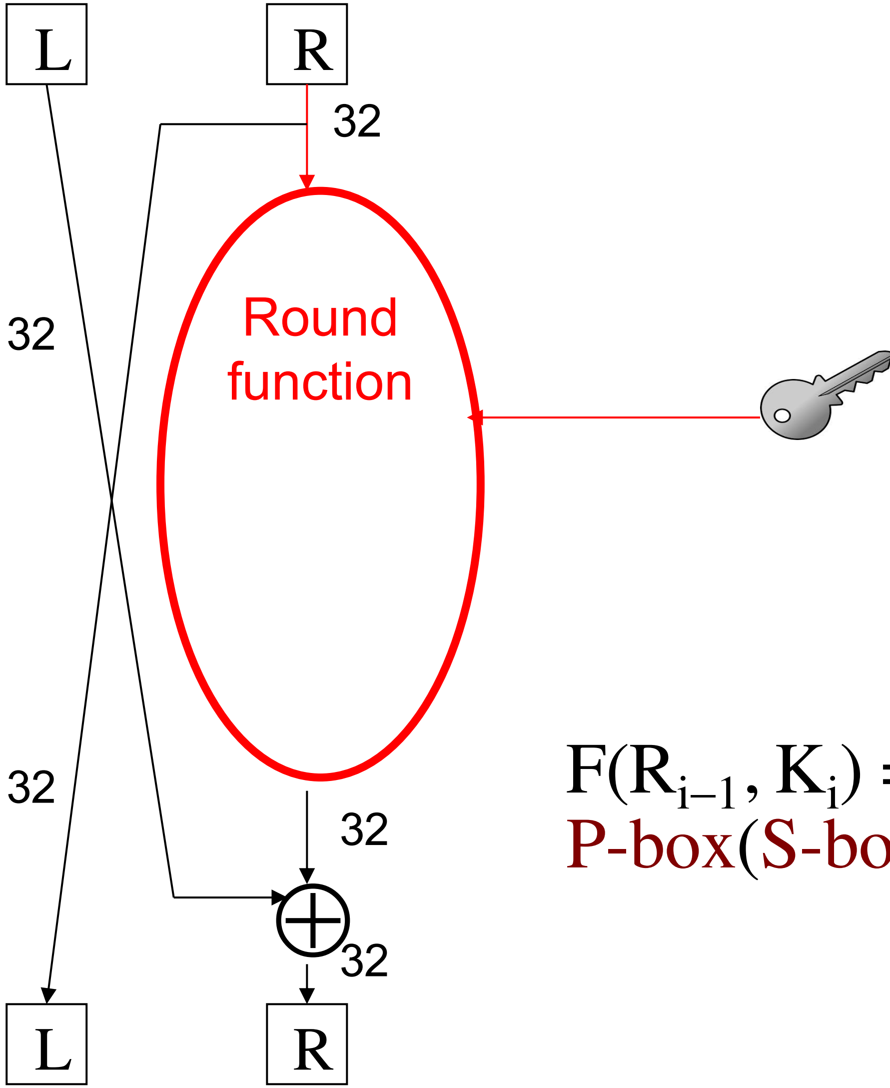
$$F(R_{i-1}, K_i) = \text{P-box}(\text{S-boxes}(\text{Expand}(R_{i-1}) \oplus K_i))$$



32 bits → 32 bits

Round function

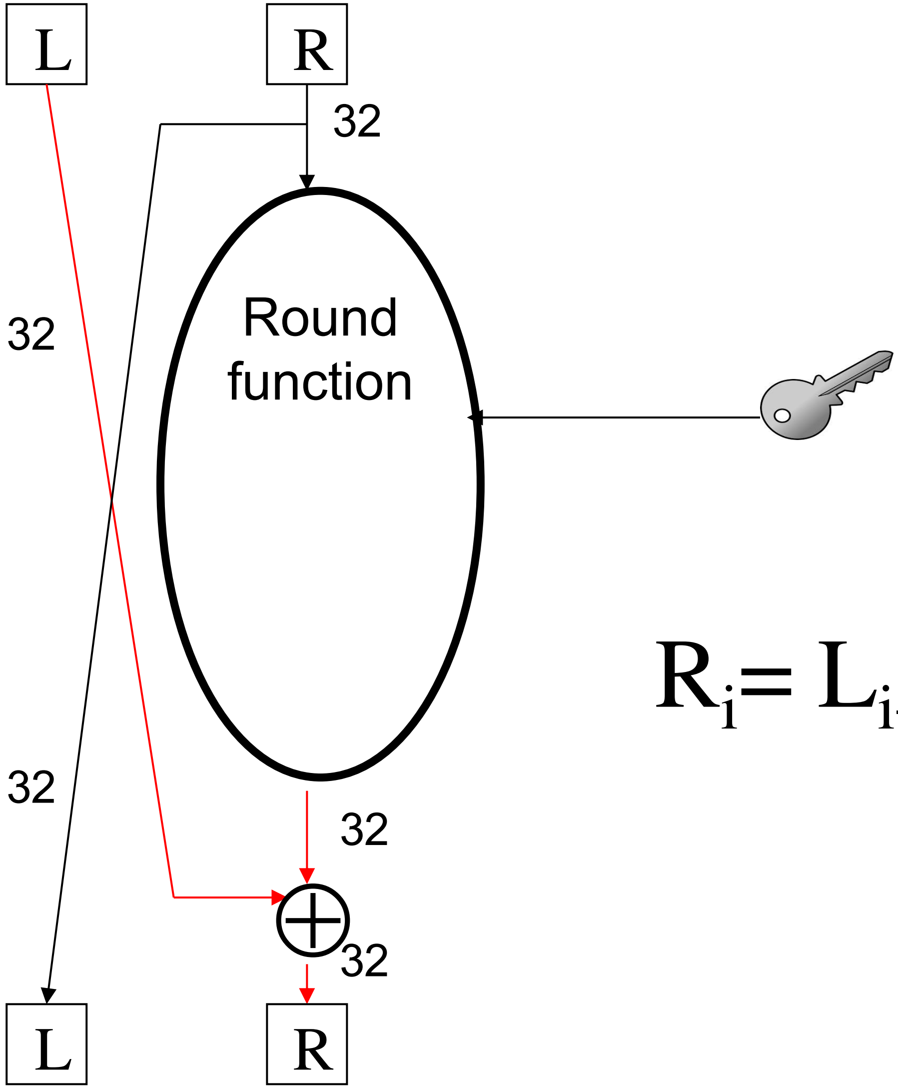
$$F(R_{i-1}, K_i) = \text{P-box}(\text{S-boxes}(\text{Expand}(R_{i-1}) \oplus K_i))$$



Round function

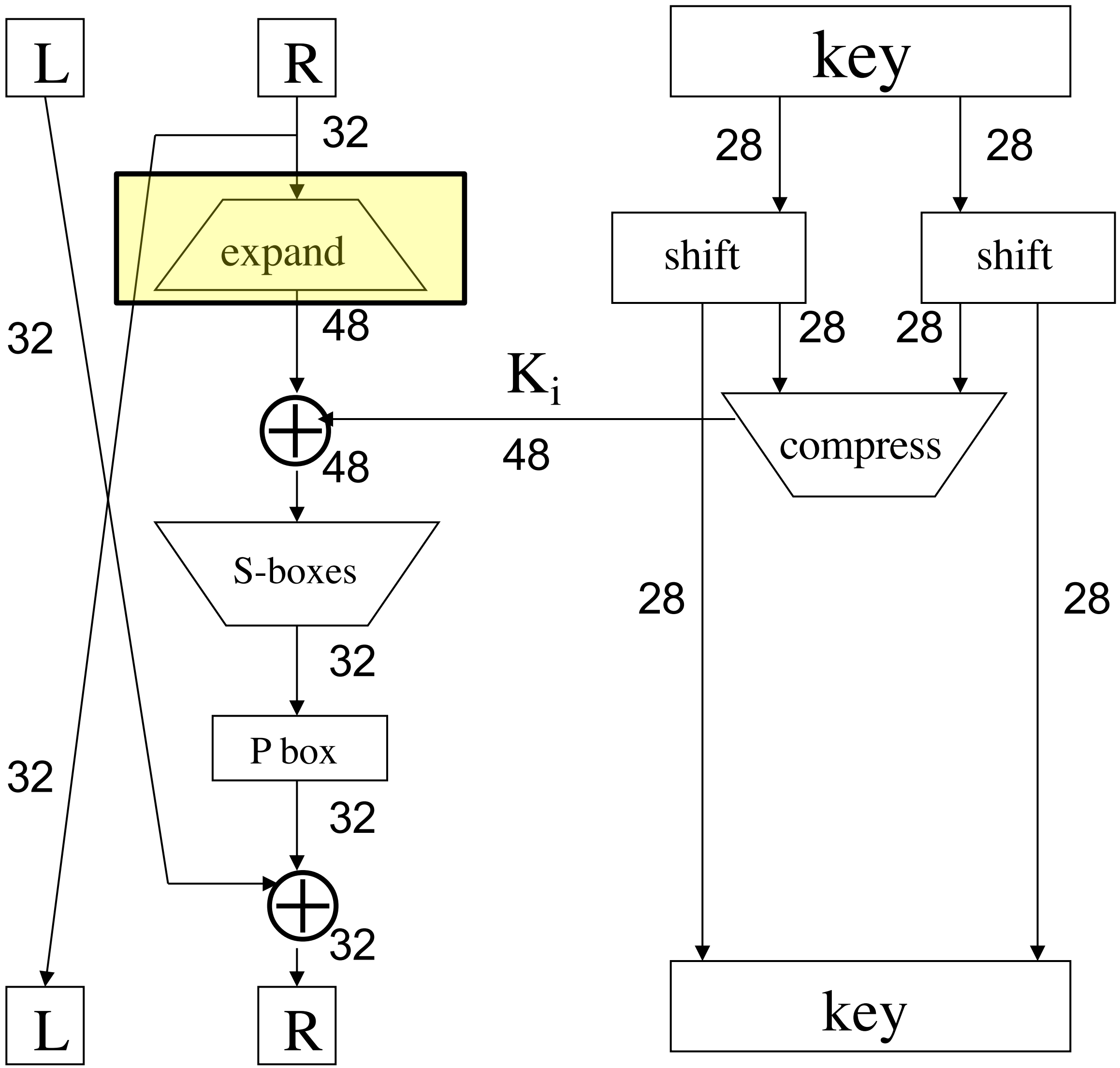
$$F(R_{i-1}, K_i) = \text{P-box}(\text{S-boxes}(\text{Expand}(R_{i-1}) \oplus K_i))$$

Step 1I in Feistel cipher



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

Expand



L
R
key
32
28
28
expand
shift
shift
48
28
28
32
K_i
compress
48
48
S-boxes
28
28
32
P box
32
32
32
L
R
key

DES Expansion

- Input 32 bits

0	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

- Output 48 bits

31	0	1	2	3	4	3	4	5	6	7	8
7	8	9	10	11	12	11	12	13	14	15	16
15	16	17	18	19	20	19	20	21	22	23	24
23	24	25	26	27	28	27	28	29	30	31	0

This mapping table is fixed

32 to 48 bits: some bits are copied twice

DES Expansion

Input 32 bits

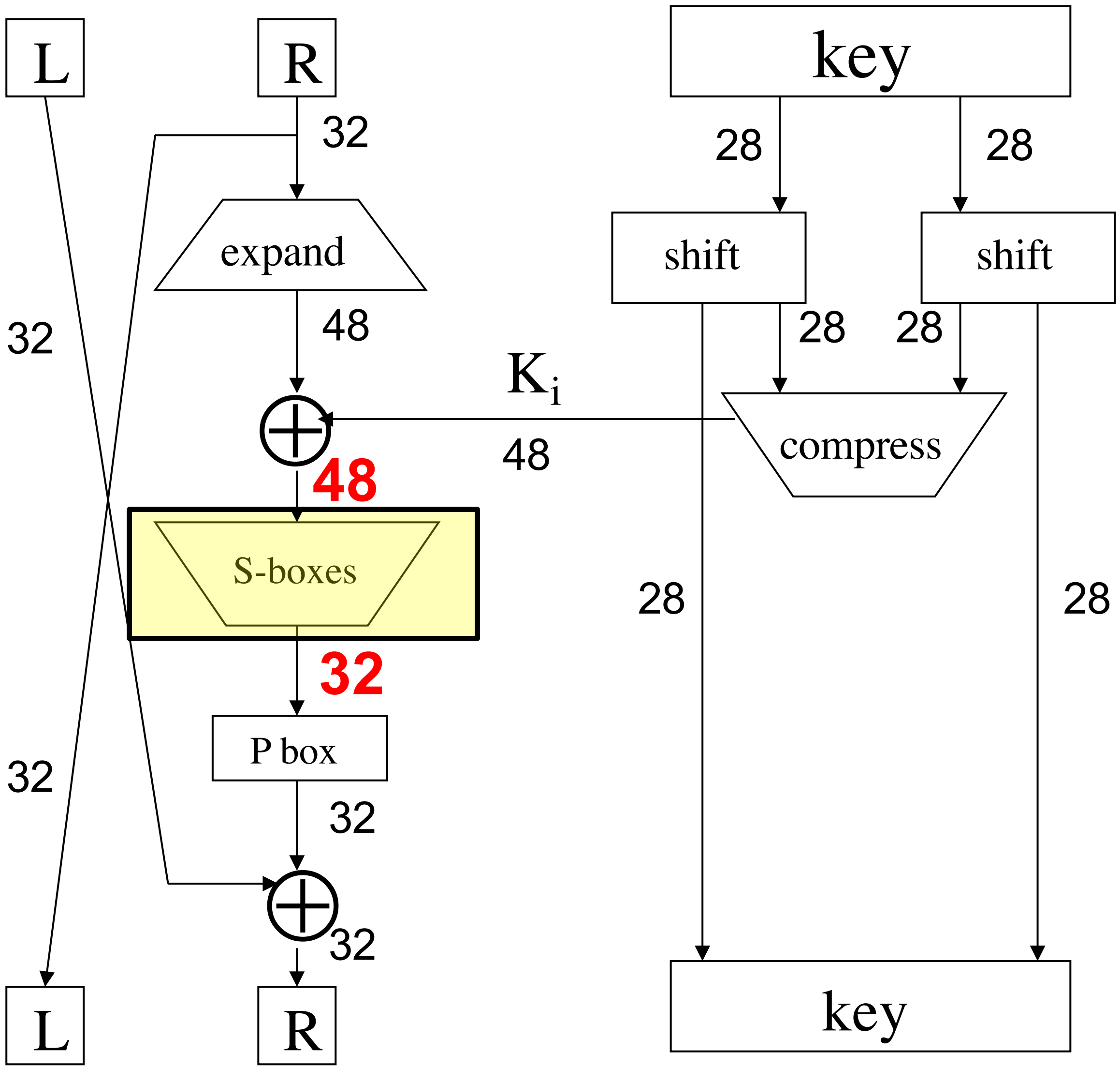
1 1 0 0 1 0 1 1 0 0 1 0 1 1 1 0
1 0 1 1 0 0 1 1 1 1 0 0 1 0 1 0

Output 48 bits

0 1 1 1 0 0 1 0 1 0 1 1
0 0 1 0 1 0 1 0 1 1 1 0
1 0 1 1 1 0 1 1 0 1 0 0
1 1 0 0 1 1 1 0 1 0 1 1

32 to 48 bits: some bits are copied twice

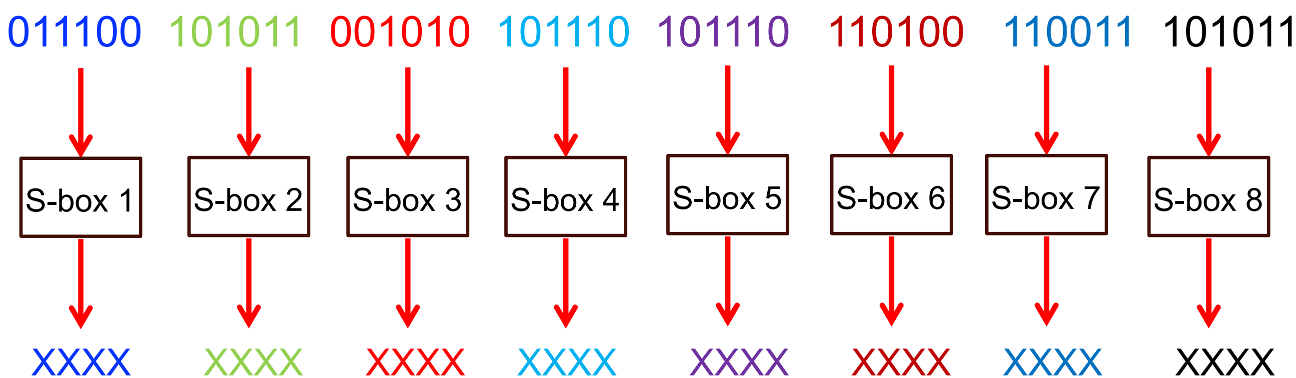
S-boxes


L
R
32
expand
48
Ki
48
48
S-boxes
32
P box
32
32
32
32
L
R
key
28
28
shift
shift
28
28
compress
28
28
key

DES S-box

- 8 different “substitution boxes” or S-boxes
- Each S-box maps 6 bits to 4 bits
- Each S-box is predefined

48 bits Input, grouped by 6 bits



32 bits output

DES S-box

Row input bits (0,5)

↓

Column input bits (1,2,3,4)

	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
00	1110	0100	1101	0001	0010	1111	1011	1000	0011	1010	0110	1100	0101	1001	0000	0111
01	0000	1111	0111	0100	1110	0010	1101	0001	1010	0110	1100	1011	1001	0101	0011	1000
10	0100	0001	1110	1000	1101	0110	0010	1011	1111	1100	1001	0111	0011	1010	0101	0000
11	1111	1100	1000	0010	0100	1001	0001	0111	0101	1011	0011	1110	1010	0000	0110	1101

Taking 011100 as input

Row(00)

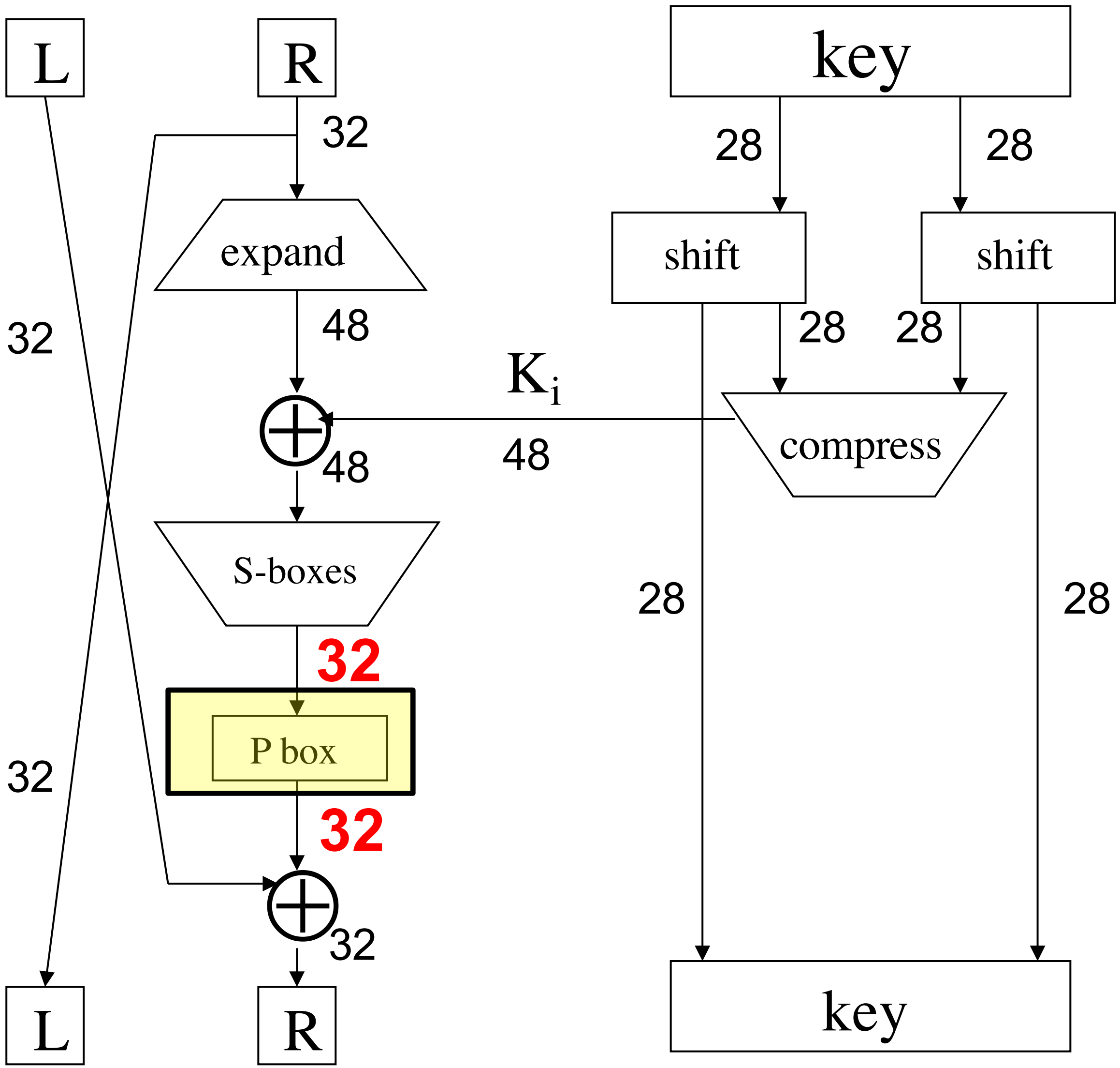
Col(1110)

01110 -> 0000

Lookup table

Binghamton University
State University of New York

P-box



L
R
key
32
28
28
expand
shift
shift
32
48
28
28
K_i
compress
48
48
S-boxes
28
28
32
P box
32
32
32
L
R
key

DES P-box(Permutation)

- **Input 32 bits**

0	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

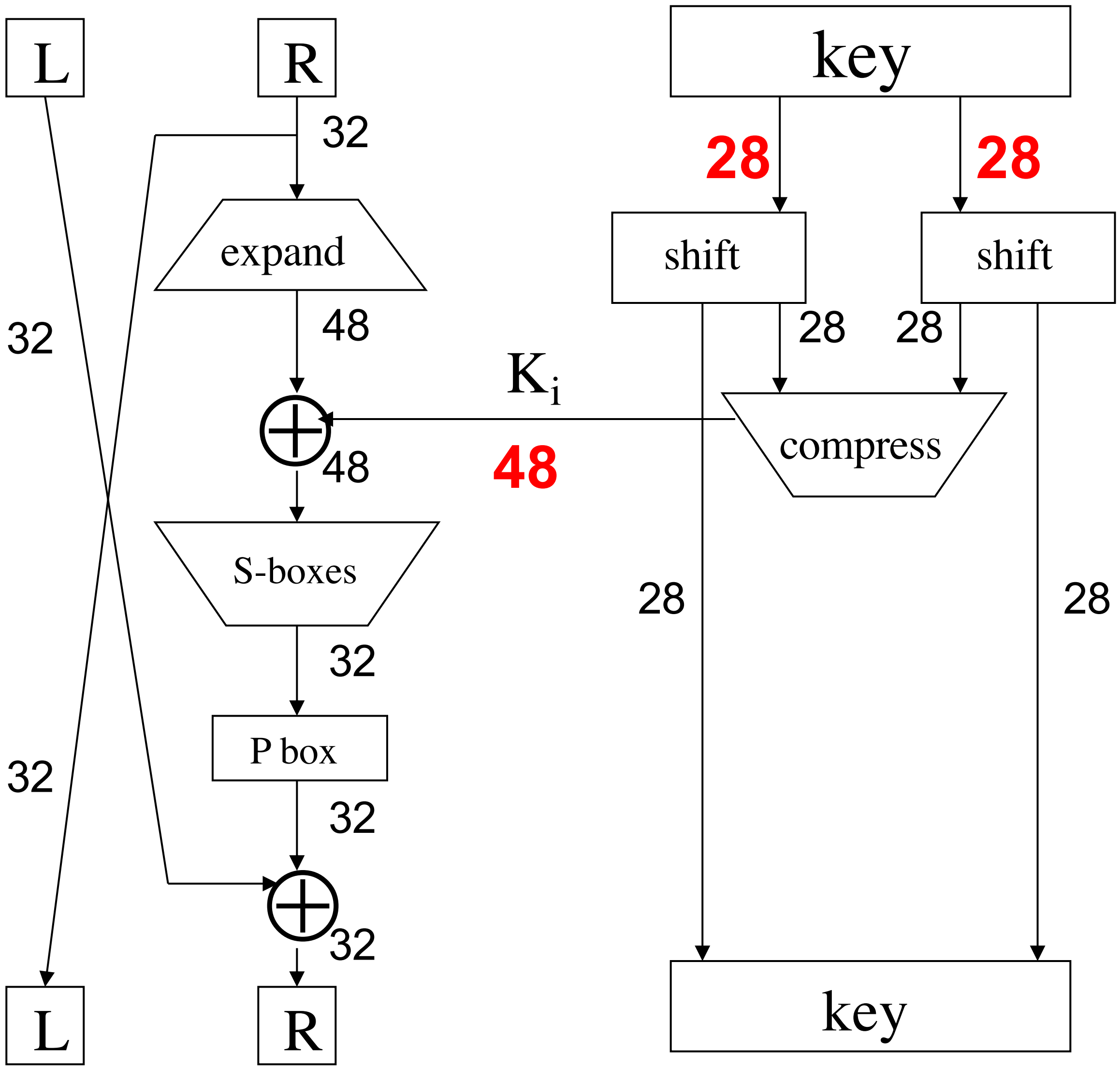
- **Output 32 bits**

15	6	19	20	28	11	27	16	0	14	22	25	4	17	30	9
1	7	23	13	31	26	2	8	18	12	29	5	21	10	3	24

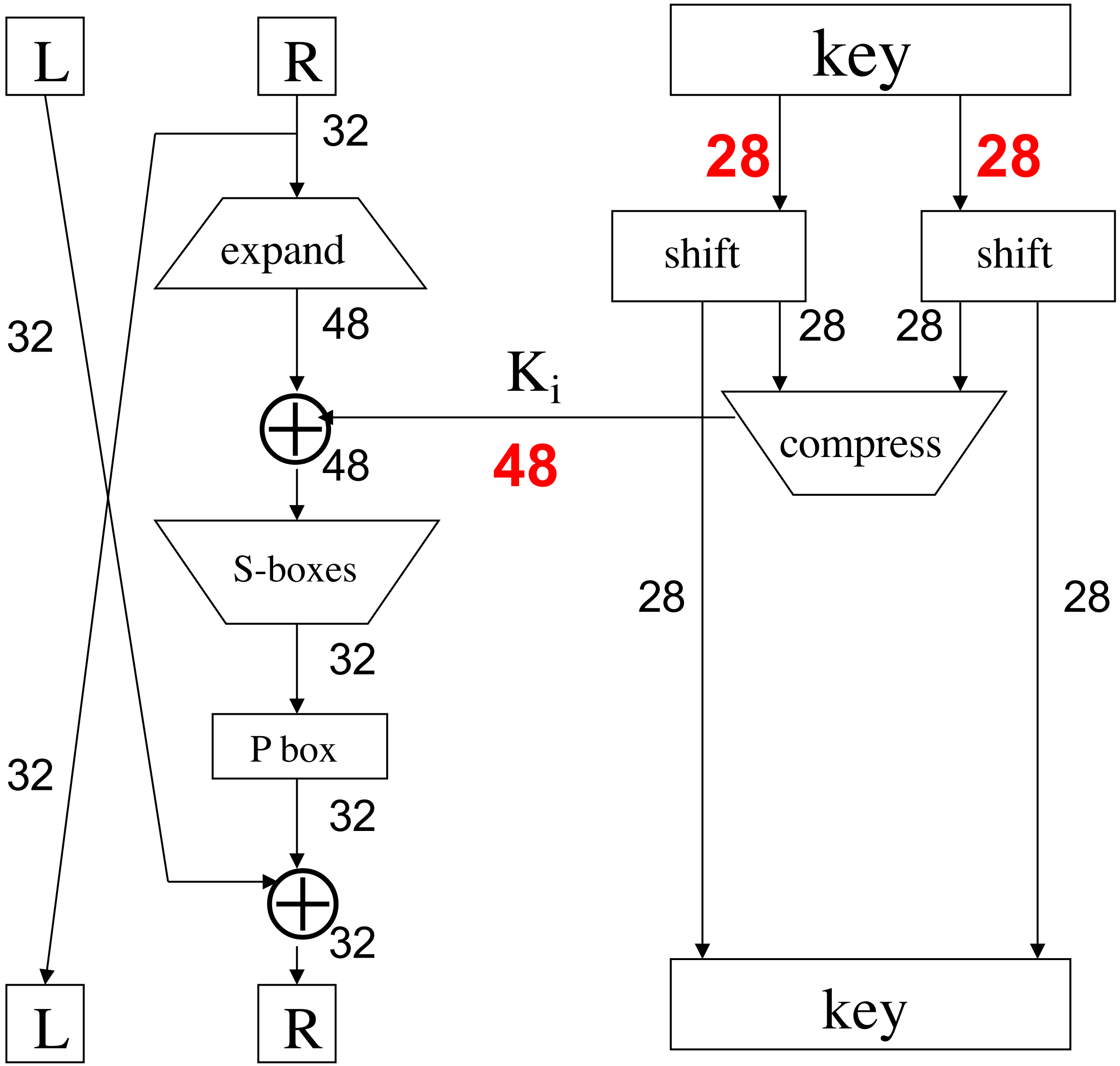
P-box is fixed

Used for permutation

Subkey

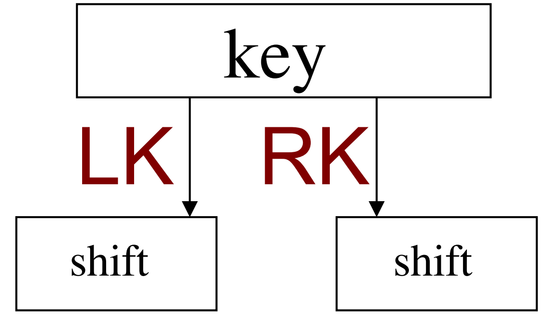


L
R
32
expand
48
32
S-boxes
32
P box
32
32
L
R
key
28
28
shift
shift
28
28
K_i
compress
48
28
28
key



Shift & Compress

Key → LK & RK



- 56 bit DES key, numbered 0,1,2,…,55
- Left half key bits, `LK`

49	42	35	28	21	14	7
0	50	43	36	29	22	15
8	1	51	44	37	30	23
16	9	2	52	45	38	31

4x7 = 28

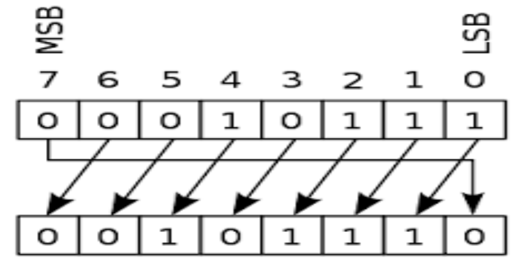
- Right half key bits, `RK`

55	48	41	34	27	20	13
6	54	47	40	33	26	19
12	5	53	46	39	32	25
18	11	4	24	17	10	3

4x7 = 28

Fixed permutation tables are used to split the key into LK, RK

DES Subkey



Circular shift left by 1

- For rounds `i=1,2,...,16`

Shift

- Let LK = (LK **circular shift left** by r_i)
- Let RK = (RK **circular shift left** by r_i)

2 x 28 = 56 bits

Compression & permutation

- Left half of subkey K_i is of LK bits

13	16	10	23	0	4	2	27	14	5	20	9
22	18	11	3	25	7	15	6	26	19	12	1

2x12 = 24 bits

- Right half of subkey K_i is RK bits

12	23	2	8	18	26	1	11	22	16	4	19
15	20	10	27	5	24	17	13	21	7	0	3

2x12 = 24 bits

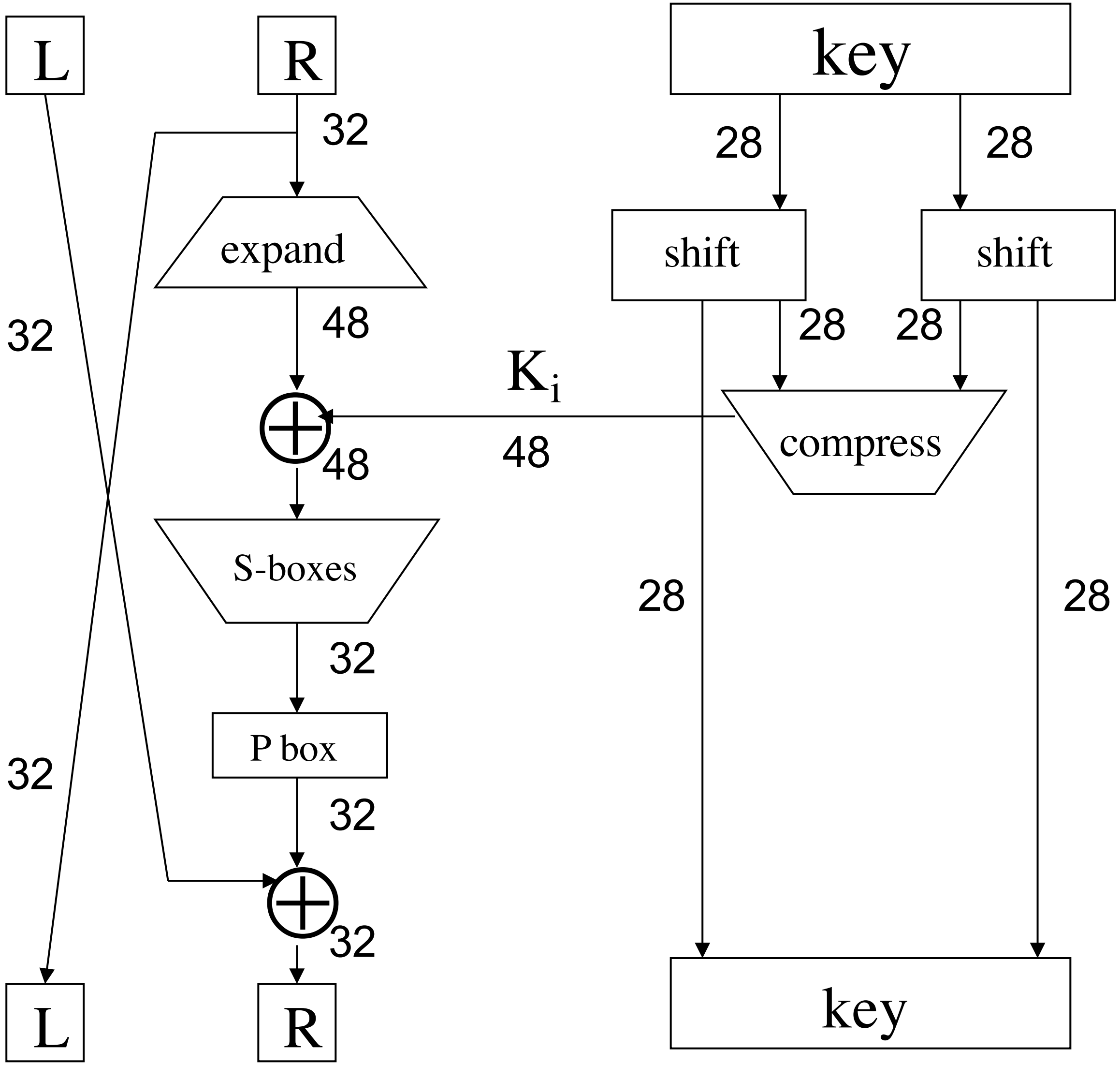
Compress: 56 → 48 bits!

DES Subkey

- For rounds $1, 2, 9$ and 16 the shift r_i is 1, and in all other rounds r_i is 2
- Bits $8, 17, 21, 24$ of LK omitted each round
- Bits $6, 9, 14, 25$ of RK omitted each round
- **Compression permutation** yields 48 bit subkey K_i from 56 bits of LK and RK
- **Key schedule** generates subkey

DES Last Word (Almost)

- An initial permutation before round 1
- Halves are swapped after last round
- A final permutation (inverse of initial perm) applied to (R_{16}, L_{16})
- None of this serves security purpose



One Round of DES

(in total: 16 rounds)

Security of DES

- Security depends heavily on S-boxes
  - Everything else in DES is **linear**
- Thirty+ years of intense analysis has revealed no "back door"
- Attacks are essentially exhaustive key search
- **Inescapable conclusions**
  - Designers of DES knew what they were doing
  - Designers of DES were way ahead of their time

Security of DES

NSA reveals its secret: No backdoor in encryption standard

By William Jackson | Feb 16, 2011

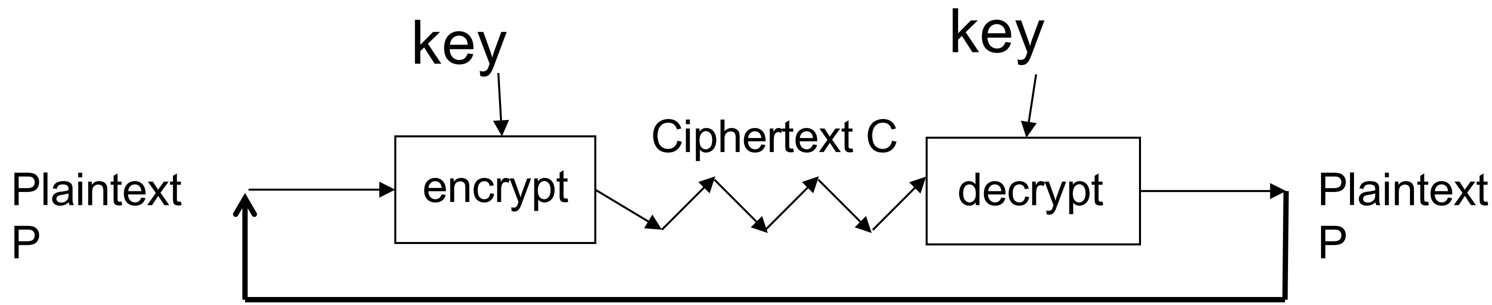
SAN FRANCISCO — The National Security Agency made changes in the proposed design of the Data Encryption Standard before its adoption in 1976, but it did not add any backdoors or other surprises that have been speculated about for 35 years, the technical director of NSA's information assurance directorate said Wednesday.
"We're actually pretty good guys," said Dickie George. "We wanted to make sure we were as squeaky clean as possible."

3-DES

Block Cipher Notation

- P = plaintext block
- C = ciphertext block

- Encrypt P with key K to get ciphertext C
  - $C = E(P, K)$
- Decrypt C with key K to get plaintext P
  - $P = D(C, K)$



- $P = D(E(P, K), K)$
- $C = E(D(C, K), K)$

- Suppose that $K_1 \neq K_2$:
  - $P \neq D(E(P, K_1), K_2)$
  - $C \neq E(D(C, K_1), K_2)$

Triple DES or 3DES

- Today, 56 bit DES key is too small
  - Exhaustive key search is feasible
- But DES is everywhere, so what to do?
- **Triple DES** or **3DES** (112 bit key)
  - $C = E(D(E(P, K_1), K_2), K_1)$
  - $P = D(E(D(C, K_1), K_2), K_1)$

Only two keys!

Triple DES or 3DES

- Today, 56 bit DES key is too small
  - Exhaustive key search is feasible
- But DES is everywhere, so what to do?
- **Triple DES** or **3DES** (112 bit key)
  - $C = E(D(E(P, K_1), K_2), K_1)$
  - $P = D(E(D(C, K_1), K_2), K_1)$

Only two keys!

$$C = E(D(E(P, K_1), K_2), K_1)$$

Triple DES or 3DES

- Today, 56 bit DES key is too small
  - Exhaustive key search is feasible
- But DES is everywhere, so what to do?
- **Triple DES** or **3DES** (112 bit key)
  - $C = E(D(E(P, K_1), K_2), K_1)$
  - $P = D(E(D(C, K_1), K_2), K_1)$

Only two keys!

$$C = E(\underline{D(E(P, K_1), K_2)}, K_1)$$

$$\Longrightarrow D(C, K_1) = \underline{D(E(P, K_1), K_2)}$$

Triple DES or 3DES

- Today, 56 bit DES key is too small
  - Exhaustive key search is feasible
- But DES is everywhere, so what to do?
- **Triple DES** or **3DES** (112 bit key)
  - $C = E(D(E(P, K_1), K_2), K_1)$
  - $P = D(E(D(C, K_1), K_2), K_1)$

Only two keys!

$$C = E(D(E(P, K_1), K_2), K_1)$$

$$\Rightarrow D(C, K_1) = D(\underline{E(P, K_1)}, K_2)$$

$$\Rightarrow E(D(C, K_1), K_2) = \underline{E(P, K1)}$$

Triple DES or 3DES

- Today, 56 bit DES key is too small
  - Exhaustive key search is feasible
- But DES is everywhere, so what to do?
- **Triple DES** or **3DES** (112 bit key)
  - $C = E(D(E(P, K_1), K_2), K_1)$
  - $P = D(E(D(C, K_1), K_2), K_1)$

Only two keys!

$$C = E(D(E(P, K_1), K_2), K_1)$$

$$\longrightarrow D(C, K_1) = D(E(P, K_1), K_2)$$

$$\longrightarrow E(D(C, K_1), K_2) = E(\underline{P}, K1)$$

$$\longrightarrow D(E(D(C, K_1), K_2), K_1) = \underline{P}$$

BINGHAMTON UNIVERSITY
STATE UNIVERSITY OF NEW YORK

Triple DES or 3DES

- Today, 56 bit DES key is too small
  - Exhaustive key search is feasible
- But DES is everywhere, so what to do?
- **Triple DES** or **3DES** (112 bit key)
  - $C = E(D(E(P, K_1), K_2), K_1)$
  - $P = D(E(D(C, K_1), K_2), K_1)$

Only two keys!

$$C = E(D(E(P, K_1), K_2), K_1)$$

$$\Rightarrow D(C, K_1) = D(E(P, K_1), K_2)$$

$$\Rightarrow E(D(C, K_1), K_2) = E(P, K1)$$

$$\Rightarrow D(E(D(C, K_1), K_2), K_1) = P$$

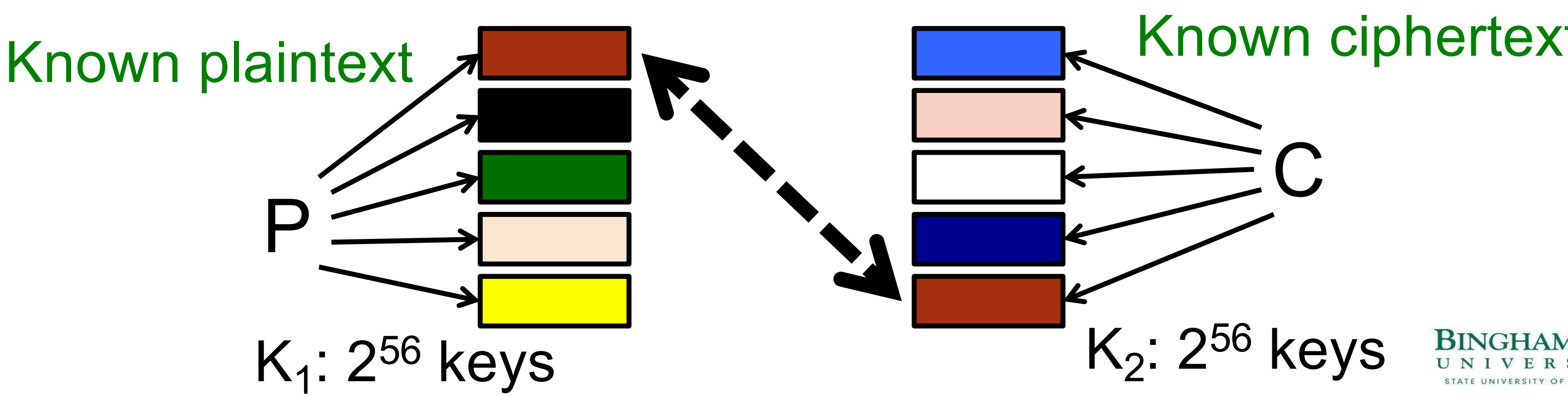
BINGHAMTON UNIVERSITY STATE UNIVERSITY OF NEW YORK

More on 3DES

- Why Encrypt-Decrypt-Encrypt with 2 keys?
  - Backward compatible: $E(D(E(P, K), K), K) = E(P, K)$
  - And 112 bits is enough
- Why not $C = E(E(P, K_1), K_2)$?
  - A (semi-practical) **known plaintext** attack

Meet-in-the-middle attack

- Pre-compute table of $\mathrm{E}(\mathrm{P}, \mathrm{K}_1)$ for every possible key K_1 (resulting table has 2^{56} entries) used for search
- Then for each possible K_2 compute $\mathrm{D}(\mathrm{C}, \mathrm{K}_2)$ until a match in table is found (2^{56})
- When match is found, have $\mathrm{E}(\mathrm{P}, \mathrm{K}_1) = \mathrm{D}(\mathrm{C}, \mathrm{K}_2)$
- Result gives us keys: $\mathrm{C} = \mathrm{E}(\mathrm{E}(\mathrm{P}, \mathrm{K}_1), \mathrm{K}_2)$



BINGHAMTON UNIVERSITY
STATE UNIVERSITY OF NEW YORK

Cost comparison

- **Brute force attack**: 2^{112}

- **Meet-in-the-middle attack**: $2^{56} + 2^{56} = 2^{57}$