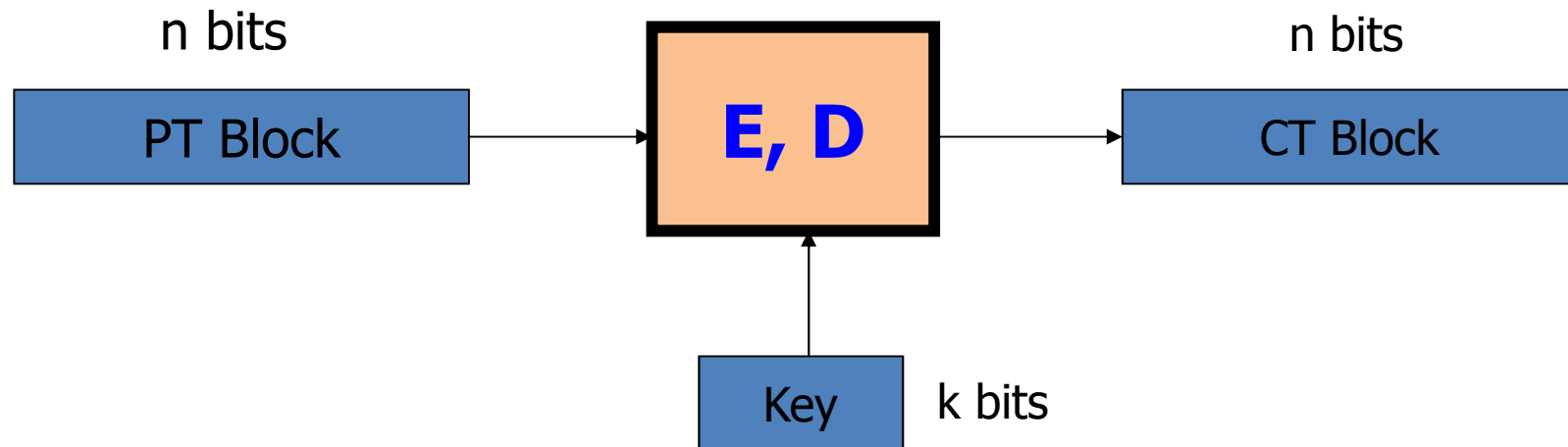


Block Ciphers



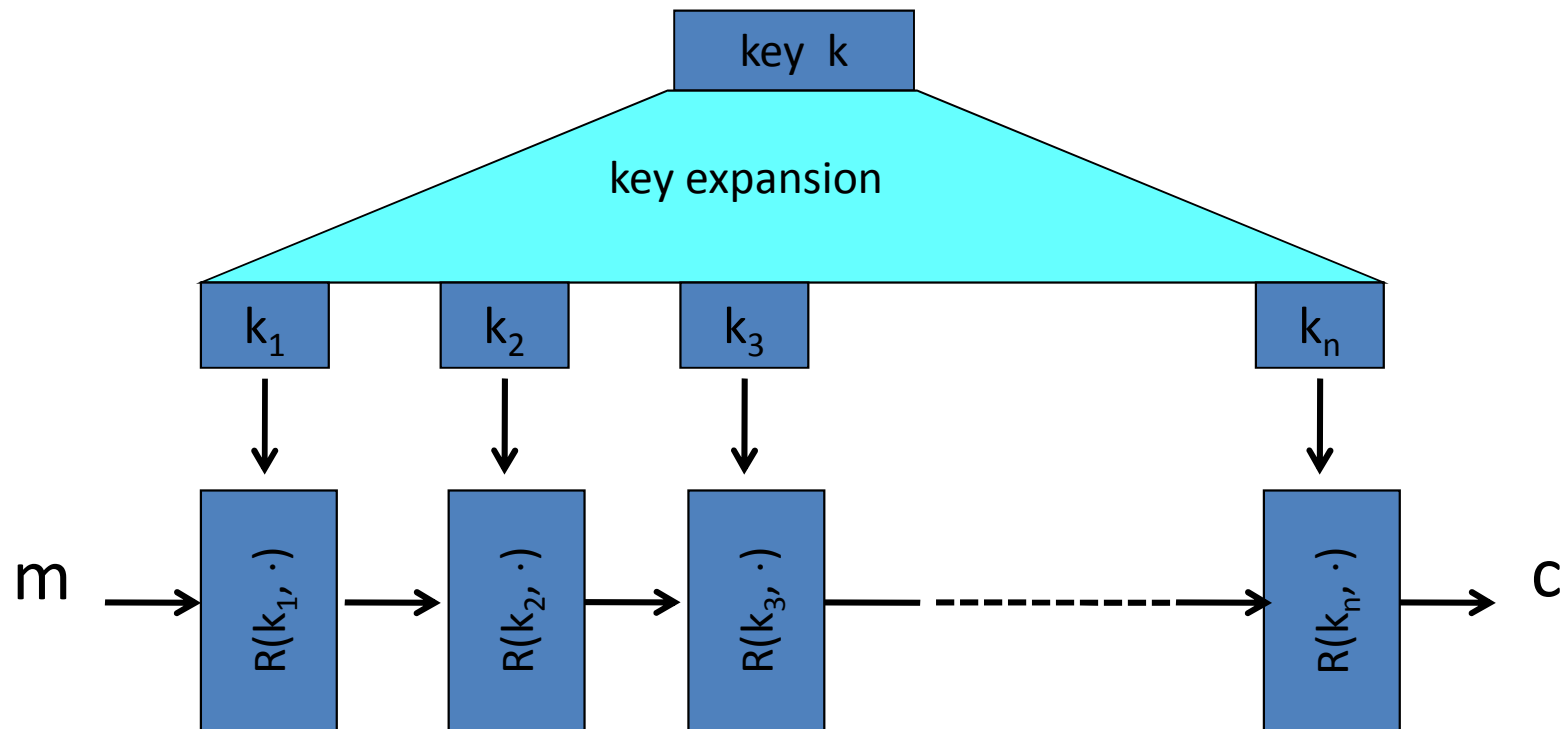
Block ciphers: crypto work horse



Canonical examples:

1. 3DES: $n = 64$ bits, $k = 168$ bits
2. AES: $n = 128$ bits, $k = 128, 192, 256$ bits

Block Ciphers Built by Iteration



$R(k, m)$ is called a round function

for 3DES ($n=48$), for AES-128 ($n=10$)

(Iterated) Block Cipher

- Plaintext and ciphertext consist of fixed-sized blocks
- 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: Encryption

- **Feistel cipher** is a type of block cipher, not a specific block cipher
- Split plaintext block into left and right halves: $P = (L_0, R_0)$
- For each round $i = 1, 2, \dots, n$, 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: $C = (L_n, R_n)$

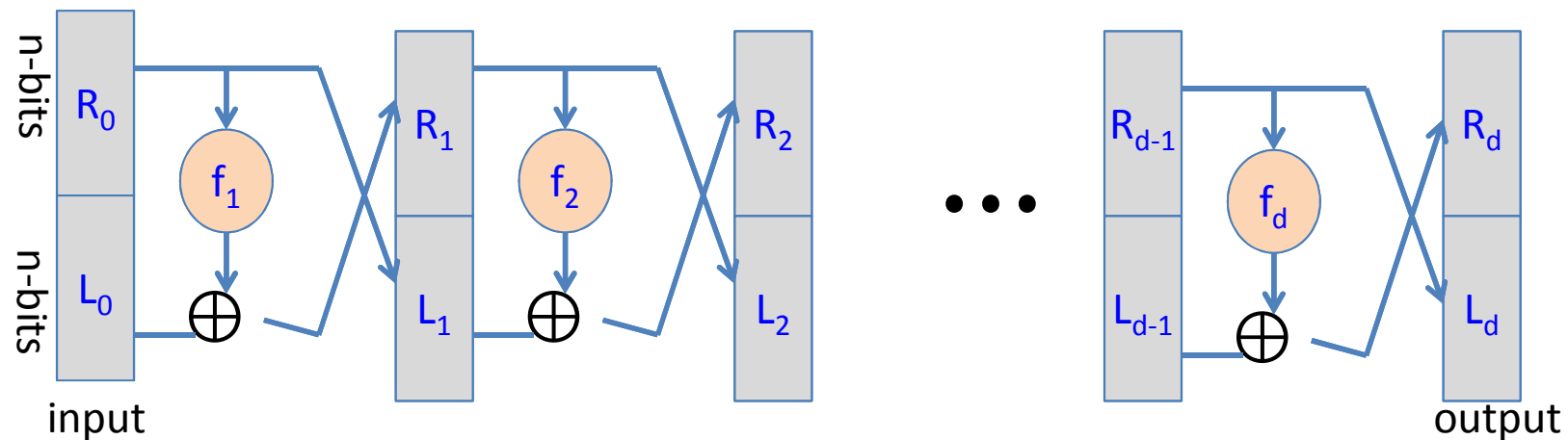
Feistel Cipher: Decryption

- Start with ciphertext $C = (L_n, R_n)$
- For each round $i = n, n-1, \dots, 1$, 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: $P = (L_0, R_0)$
- Formula “works” for any function F
 - But only secure for certain functions F

DES: core idea – Feistel Network

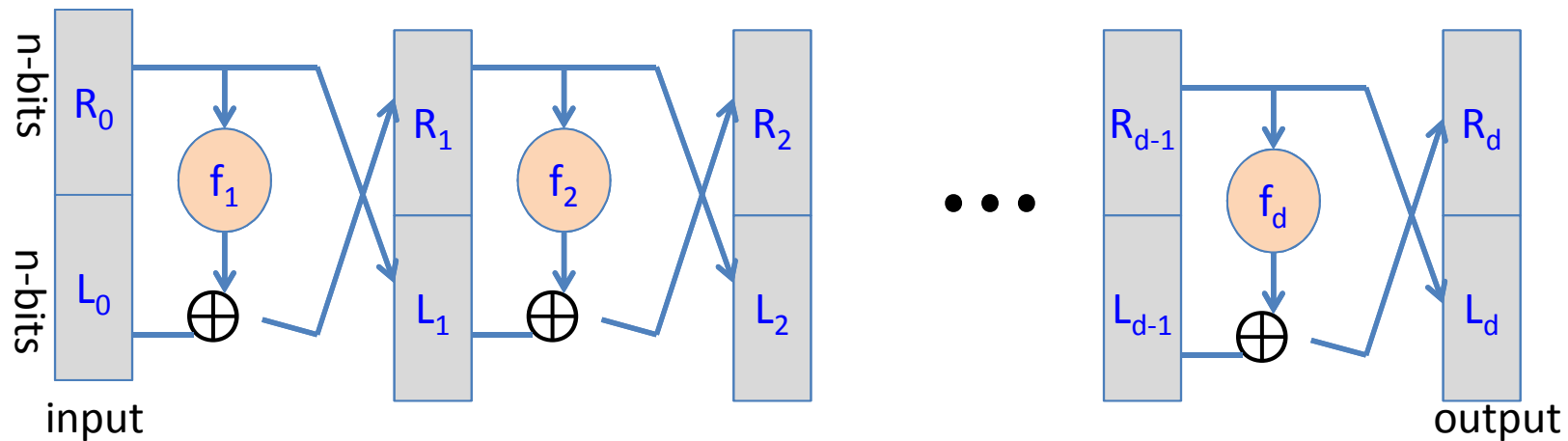
Given functions $f_1, \dots, f_d: \{0,1\}^n \rightarrow \{0,1\}^n$

Goal: build invertible function $F: \{0,1\}^{2n} \rightarrow \{0,1\}^{2n}$



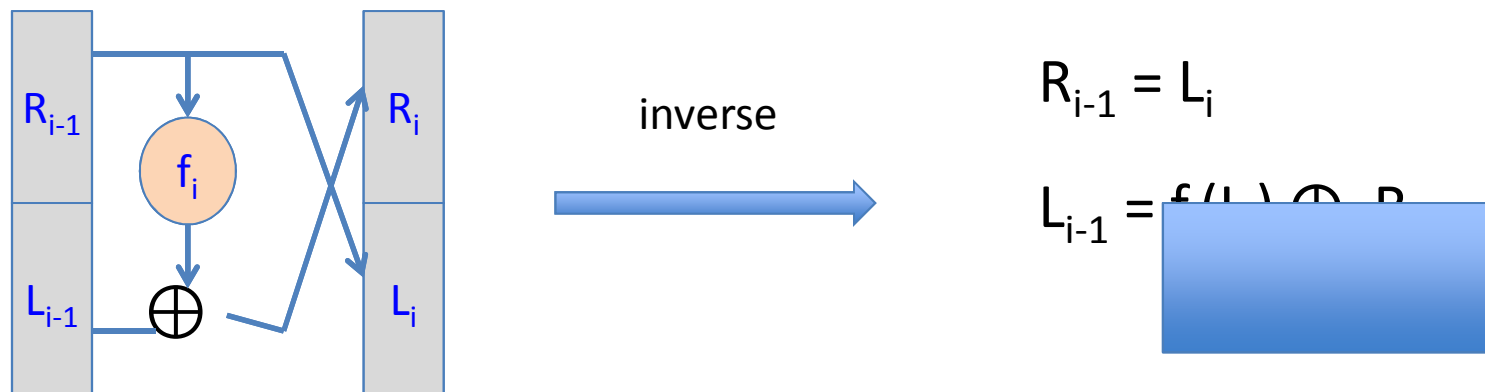
In symbols:

$$\begin{cases} R_i = f_i(R_{i-1}) \oplus L_{i-1} \\ L_i = R_{i-1} \end{cases}$$



Claim: for all $f_1, \dots, f_d: \{0,1\}^n \rightarrow \{0,1\}^n$
 Feistel network $F: \{0,1\}^{2n} \rightarrow \{0,1\}^{2n}$ is
 invertible

Proof: construct inverse



Data Encryption Standard

- **DES** developed in 1970's
- Based on IBM's Lucifer cipher
- DES was U.S. government standard
- DES development was controversial
 - NSA secretly involved
 - Design process was secret
 - Key length reduced from 128 to 56 bits
 - Subtle changes to Lucifer algorithm

The Data Encryption Standard (DES)

- Early 1970s: Horst Feistel designs Lucifer at IBM
key-len = 128 bits ; block-len = 128 bits
- 1973: NBS asks for block cipher proposals.
IBM submits variant of Lucifer.
- 1976: NBS adopts DES as a federal standard
key-len = 56 bits ; block-len = 64 bits
- 1997: DES broken by exhaustive search
- 2000: NIST adopts Rijndael as AES to replace DES

Widely deployed in banking (ACH) and commerce

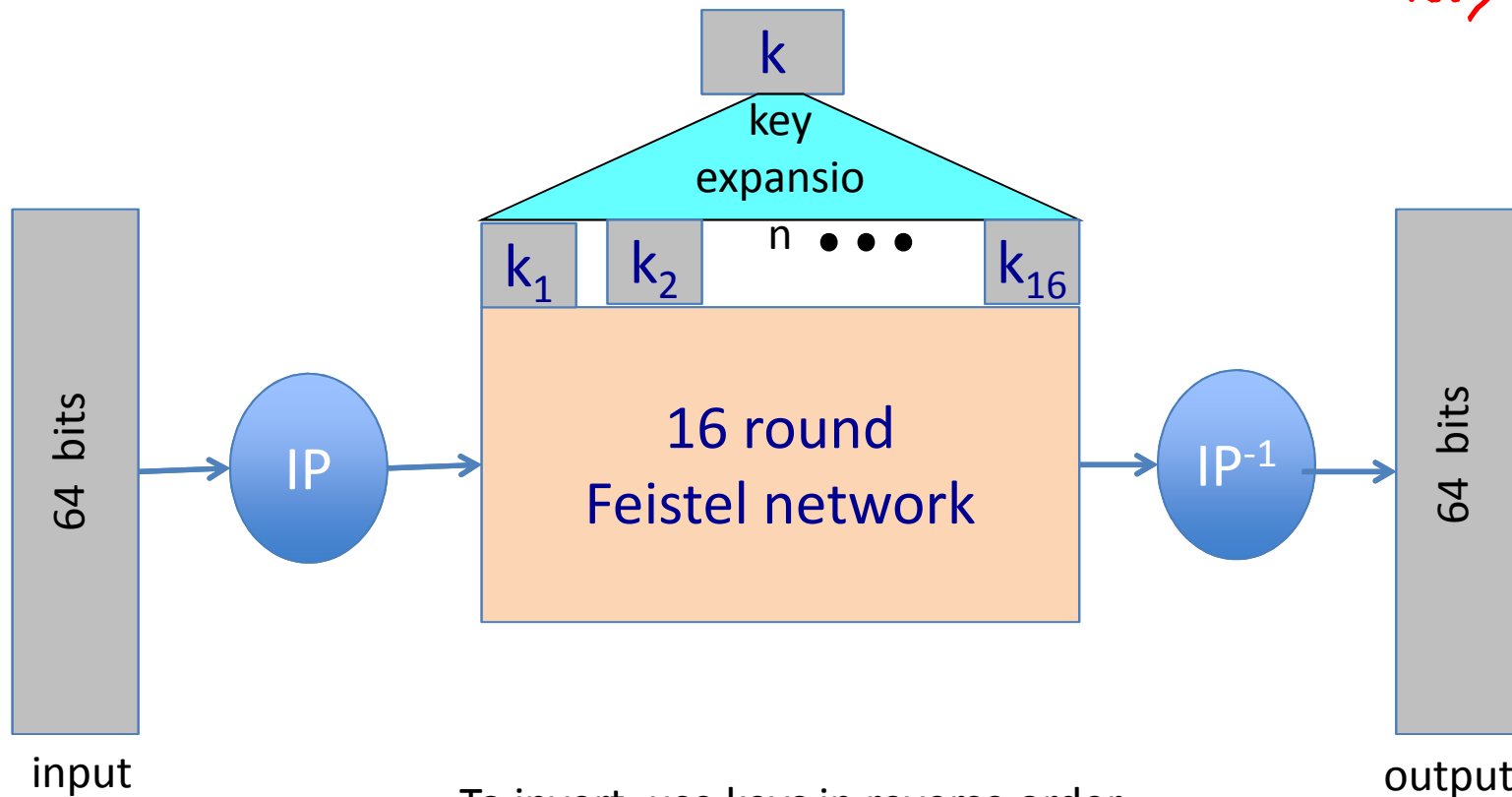
DES Numerology

- 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”
 - Each S-boxes maps 6 bits to 4 bits

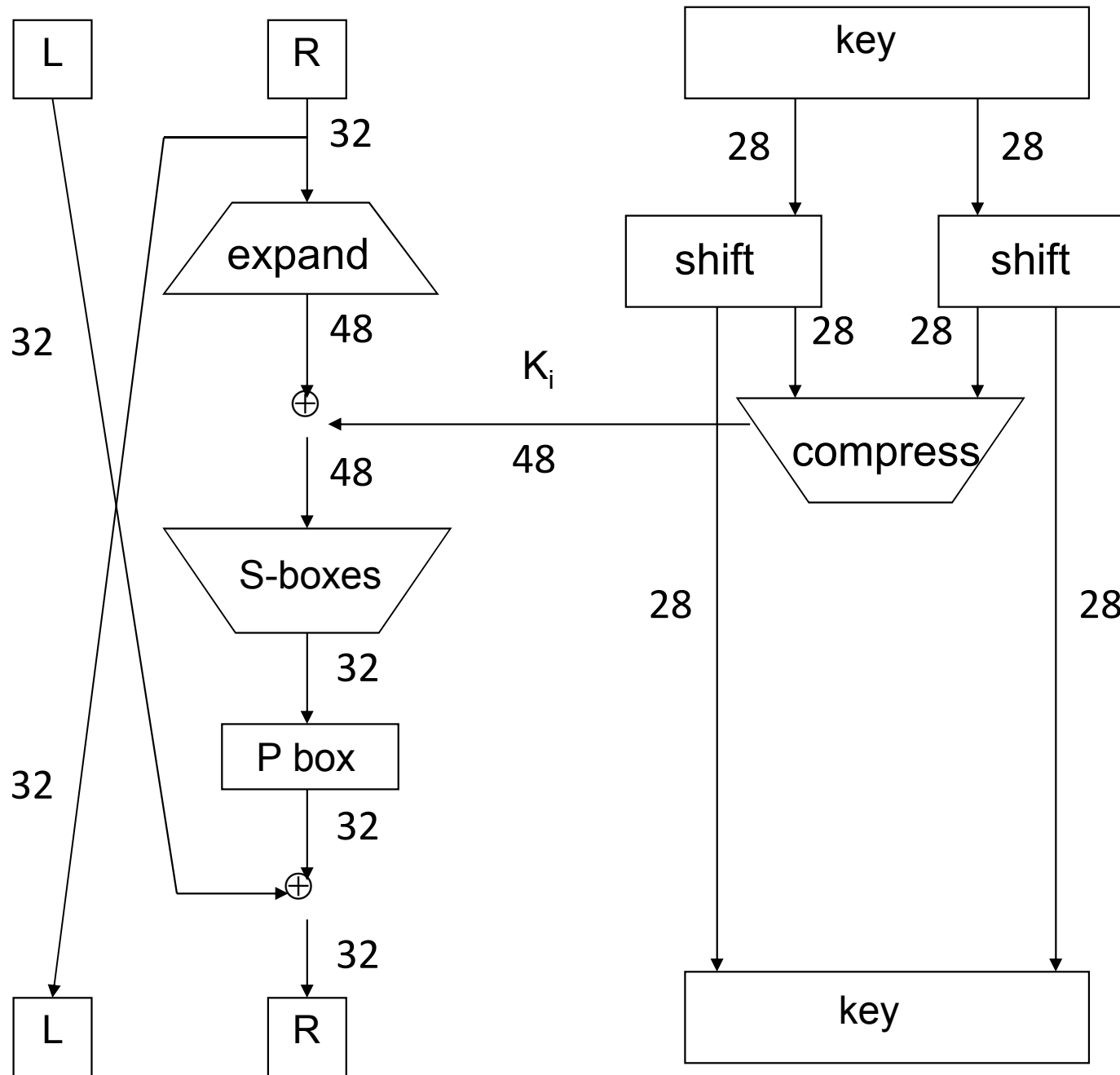
DES: 16 round Feistel network

$$f_1, \dots, f_{16}: \{0,1\}^{32} \rightarrow \{0,1\}^{32} \quad , \quad f_i(x) = F(k_i, x)$$

*From
key k*



To invert, use keys in reverse order



One
Round
of
DES

DES Expansion 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 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

DES S-box

- 8 “substitution boxes” or S-boxes
- Each S-box maps 6 bits to 4 bits
- S-box number 1

input bits (0,5)

↓

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

DES P-box

- 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

DES Subkey

- 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

- 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

DES Subkey

- For rounds $i=1, 2, \dots, 16$
 - Let $LK = (LK \text{ circular shift left by } r_i)$
 - Let $RK = (RK \text{ circular shift left by } r_i)$
 - 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

- 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

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

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, essentially exhaustive key search
- **Inescapable conclusions**
 - Designers of DES knew what they were doing
 - Designers of DES were way ahead of their time

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
 - $P = D(C, K)$
- **Note:** $P = D(E(P, K), K)$ and $C = E(D(C, K), K)$
 - But $P \neq D(E(P, K_1), K_2)$ and $C \neq E(D(C, K_1), K_2)$ when $K_1 \neq K_2$

Triple DES

- 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)
 - $C = E(D(E(P, K_1), K_2), K_1)$
 - $P = D(E(D(C, K_1), K_2), K_1)$
- Why Encrypt-Decrypt-Encrypt with 2 keys?
 - Backward compatible: $E(D(E(P, K), K), K) = E(P, K)$
 - And 112 bits is enough

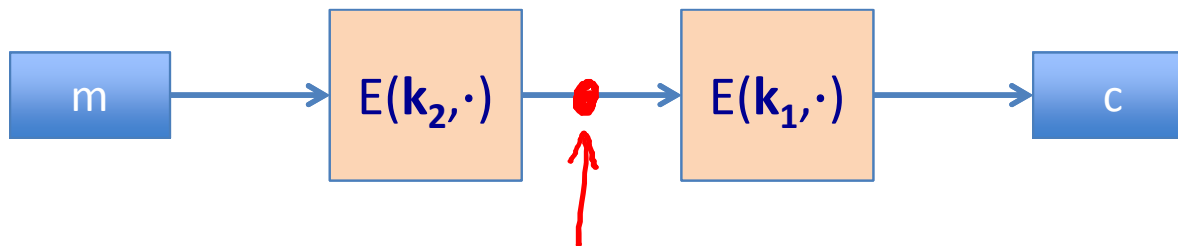
3DES

- Why not $C = E(E(P, K), K)$?
 - Trick question --- it's still just 56 bit key
- Why not $C = E(E(P, K_1), K_2)$?
- A (semi-practical) **known plaintext** attack
 - Pre-compute table of $E(P, K_1)$ for every possible key K_1 (resulting table has 2^{56} entries)
 - Then for each possible K_2 compute $D(C, K_2)$ until a match in table is found
 - When match is found, have $E(P, K_1) = D(C, K_2)$
 - Result gives us keys: $C = E(E(P, K_1), K_2)$

Why not double DES?

- Define $2E((k_1, k_2), m) = E(k_1, E(k_2, m))$

key-len = 112 bits for DES



Attack: $M = (m_1, \dots, m_{10})$, $C = (c_1, \dots, c_{10})$.

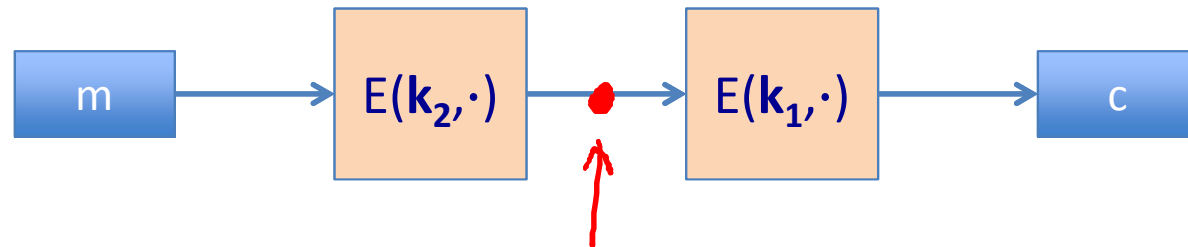
Find (K_1, K_2) s.t.
 $E(K_1, E(K_2, M)) = C$
 Equivalently:
 $E(K_2, M) = D(K_1, C)$

- step 1: build table.

sort on 2nd column

$k^0 = 00\dots00$	$E(k^0, M)$	} 2^{56} entries
$k^1 = 00\dots01$	$E(k^1, M)$	
$k^2 = 00\dots10$	$E(k^2, M)$	
$\vdots$	$\vdots$	
$k^N = 11\dots11$	$E(k^N, M)$	

Meet in the middle attack

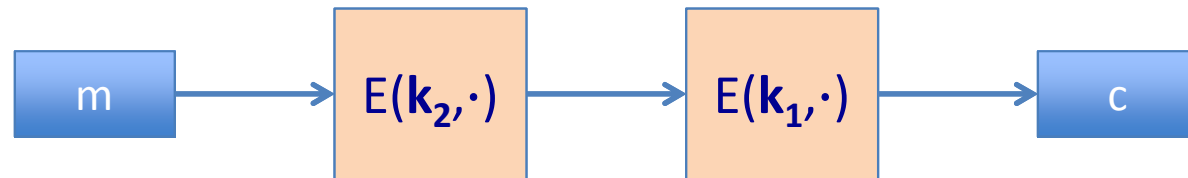


Attack: $M = (m_1, \dots, m_{10})$, $C = (c_1, \dots, c_{10})$

$k^0 = 00\dots00$	$E(k^0, M)$
$k^1 = 00\dots01$	$E(k^1, M)$
$k^2 = 00\dots10$	$E(k^2, M)$
$\vdots$	$\vdots$
$k^N = 11\dots11$	$E(k^N, M)$

- step 1: build table.
- Step 2: for all $k \in \{0,1\}^{56}$ do:
 - test if $D(k, C)$ is in 2nd column.
 - if so then $E(k^i, M) = D(k, C) \Rightarrow (k^i, k) = (k_2, k_1)$

Meet in the middle attack



$$\text{Time} = 2^{56} \log(2^{56}) + 2^{56} \log(2^{56}) < 2^{63} \ll 2^{112},$$

space $\approx 2^{56}$

build + sort table *search in table*

Same attack on 3DES: Time = 2^{118} , space $\approx 2^{56}$

