# Topic 1.2
# Symmetric encryption – Block ciphers

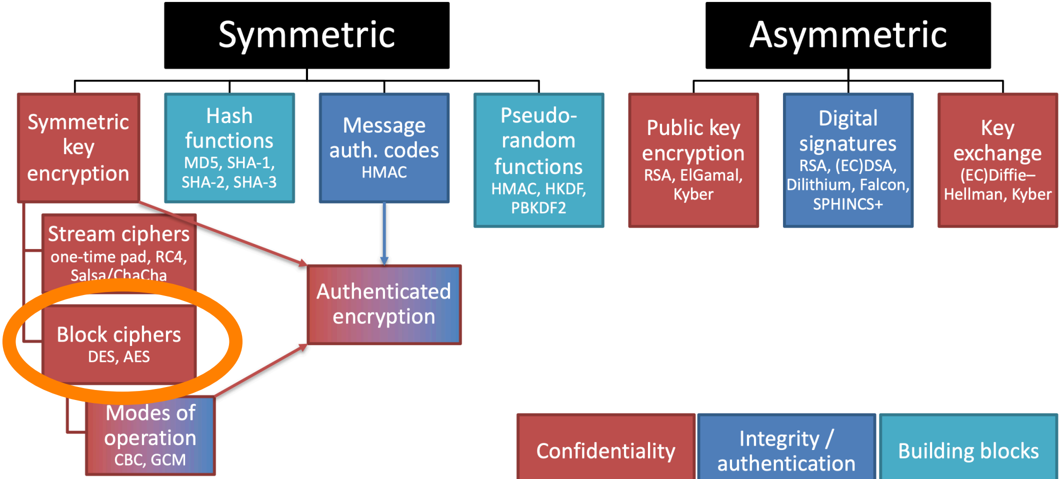
Douglas Stebila

CO 487/687: Applied Cryptography

Fall 2024

UNIVERSITY OF WATERLOO

# Map of cryptographic primitives



Overview of block ciphers

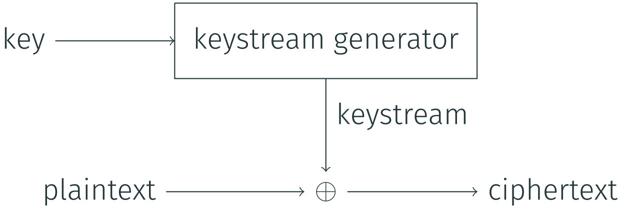
Advanced Encryption Standard (AES)

Data Encryption Standard (DES)

- Feistel networks
- Construction of DES
- Problems with DES
- Trying to save DES: Multiple encryption

# Outline

# Stream ciphers

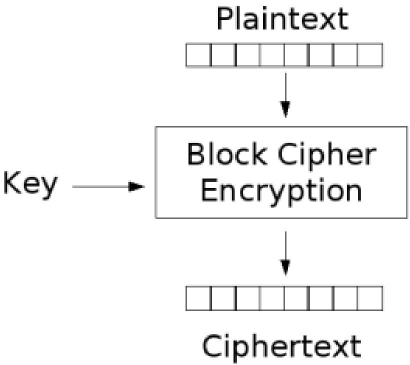



A stream cipher is a symmetric-key encryption scheme in which each successive character of plaintext determines a single character of ciphertext.

Examples:

- Substitution cipher
- Vigenère cipher
- One-time pad
- Enigma

# Block ciphers




A block cipher is a symmetric-key encryption scheme in which a fixed-length block of plaintext determines an equal-sized block of ciphertext.

Examples:

- DES
- AES

# Some Desirable Properties of Block Ciphers

Design principles described by Claude Shannon in 1949:

- Security:
  - Diffusion: each ciphertext bit should depend on all plaintext and all key bits.
  - Confusion: the relationship between key bits, plaintext bits, and ciphertext bits should be complicated.
  - Cascade or avalanche effect: changing one bit of plaintext or key should change each bit of ciphertext with probability about 50%
  - Key length: should be small, but large enough to preclude exhaustive key search.
- Efficiency:
  - Simplicity (easier to implement and analyze).
  - High encryption and decryption rate.
  - Suitability for hardware or software.

# Outline

Overview of block ciphers

Advanced Encryption Standard (AES)

Data Encryption Standard (DES)

- Feistel networks
- Construction of DES
- Problems with DES
- Trying to save DES: Multiple encryption

# The Advanced Encryption Standard (AES)

- `www.nist.gov/aes`
- Sept. 1997: Call issued for AES candidate algorithms.
- Requirements:
  - Key sizes: 128, 192 and 256 bits.
  - Block size: 128 bits.
  - Efficient on both hardware and software platforms.
  - Availability on a worldwide, non-exclusive, royalty-free basis.

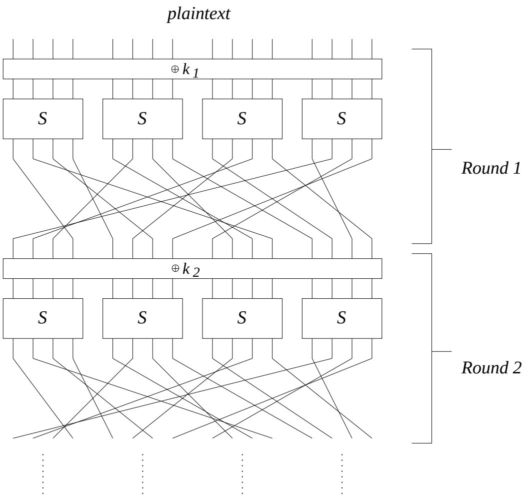
# The AES Standardization Process

- Aug. 1998: 15 submissions in Round 1.
- Aug. 1999: NIST selects five finalists:
  - MARS, RC6, Rijndael, Serpent, Twofish.
- 1999: NSA performs a hardware efficiency comparison.
- Oct. 2, 2000: Rijndael is selected.
- Dec. 2001: The AES standard is officially adopted (FIPS 197).
- Rijndael is an iterated block cipher, based on a substitution-permutation network design.

# Substitution-Permutation Networks

A substitution-permutation network (SPN) is a multiple-round iterated block cipher where each round consists of a substitution operation followed by a permutation operation.

During each round, a round key is XORed into the state. The round keys $k_i$ are derived from the main key $k$ using a key schedule function.



# Advanced Encryption Standard

- AES is an SPN where the "permutation" operation consists of two linear transformations (one of which is a permutation).
- All operations are byte oriented.
- The block size of AES is 128 bits.
- Each round key is 128 bits.
    - A key schedule is used to generate the round keys.
- AES accepts three different key lengths. The number of rounds depends on the key length:

| key length | number of rounds $h$ |
|---|---|
| 128 | 10 |
| 192 | 12 |
| 256 | 14 |

# General structure of the cipher

As with the previous ciphers we have studied:

- The substitution operation (S-box) is the only non-linear component of the cipher.
- The permutation operations (permutation and linear transformation) spread out the non-linearities in each round.

# AES Round Operations

- Each round updates a variable called State which consists of a $4 \times 4$ array of bytes (note: $4 \cdot 4 \cdot 8 = 128$, the block size).
- State is initialized with the 128-bit plaintext:

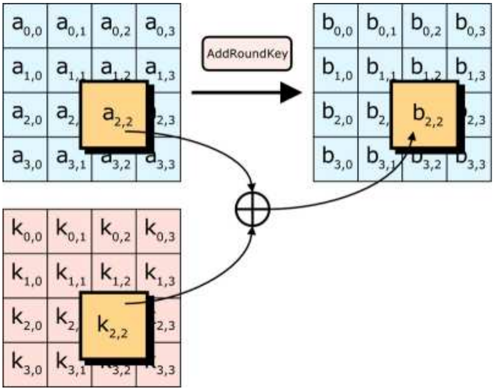
| | | | |
|---|---|---|---|
| $a_{0,0}$ | $a_{0,1}$ | $a_{0,2}$ | $a_{0,3}$ |
| $a_{1,0}$ | $a_{1,1}$ | $a_{1,2}$ | $a_{1,3}$ |
| $a_{2,0}$ | $a_{2,1}$ | $a_{2,2}$ | $a_{2,3}$ |
| $a_{3,0}$ | $a_{3,1}$ | $a_{3,2}$ | $a_{3,3}$ |

$\longleftarrow$ *plaintext*

- After $h$ rounds are completed, one final additional round key is XOR-ed with State to produce the ciphertext (key whitening).
- The AES round function uses four operations:
  - `AddRoundKey` (key mixing)
  - `SubBytes` (S-box)
  - `ShiftRows` (permutation)
  - `MixColumns` (matrix multiplication / linear transformation)

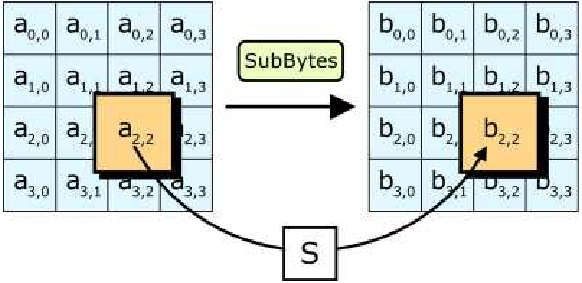
# Add Round Key

Bitwise-XOR each byte of State with the corresponding byte of the round key.



# Substitute Bytes

Take each byte in State and replace it with the output of the S-box.



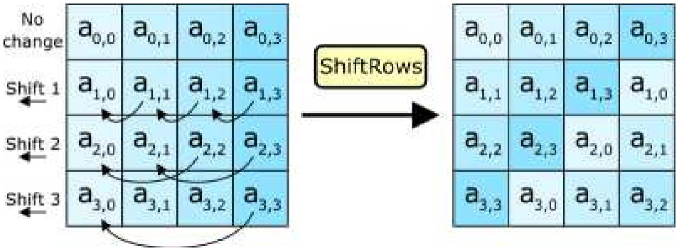

$S : \{0,1\}^8 \to \{0,1\}^8$ is a fixed and public function.

# The AES S-box

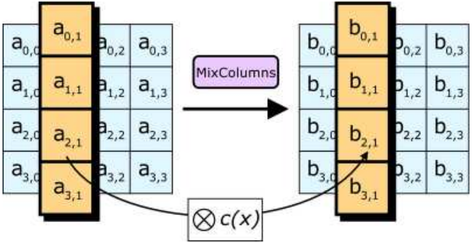
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | 63 | 7c | 77 | 7b | f2 | 6b | 6f | c5 | 30 | 01 | 67 | 2b | fe | d7 | ab | 76 |
| 10 | ca | 82 | c9 | 7d | fa | 59 | 47 | f0 | ad | d4 | a2 | af | 9c | a4 | 72 | c0 |
| 20 | b7 | fd | 93 | 26 | 36 | 3f | f7 | cc | 34 | a5 | e5 | f1 | 71 | d8 | 31 | 15 |
| 30 | 04 | c7 | 23 | c3 | 18 | 96 | 05 | 9a | 07 | 12 | 80 | e2 | eb | 27 | b2 | 75 |
| 40 | 09 | 83 | 2c | 1a | 1b | 6e | 5a | a0 | 52 | 3b | d6 | b3 | 29 | e3 | 2f | 84 |
| 50 | 53 | d1 | 00 | ed | 20 | fc | b1 | 5b | 6a | cb | be | 39 | 4a | 4c | 58 | cf |
| 60 | d0 | ef | aa | fb | 43 | 4d | 33 | 85 | 45 | f9 | 02 | 7f | 50 | 3c | 9f | a8 |
| 70 | 51 | a3 | 40 | 8f | 92 | 9d | 38 | f5 | bc | b6 | da | 21 | 10 | ff | f3 | d2 |
| 80 | cd | 0c | 13 | ec | 5f | 97 | 44 | 17 | c4 | a7 | 7e | 3d | 64 | 5d | 19 | 73 |
| 90 | 60 | 81 | 4f | dc | 22 | 2a | 90 | 88 | 46 | ee | b8 | 14 | de | 5e | 0b | db |
| a0 | e0 | 32 | 3a | 0a | 49 | 06 | 24 | 5c | c2 | d3 | ac | 62 | 91 | 95 | e4 | 79 |
| b0 | e7 | c8 | 37 | 6d | 8d | d5 | 4e | a9 | 6c | 56 | f4 | ea | 65 | 7a | ae | 08 |
| c0 | ba | 78 | 25 | 2e | 1c | a6 | b4 | c6 | e8 | dd | 74 | 1f | 4b | bd | 8b | 8a |
| d0 | 70 | 3e | b5 | 66 | 48 | 03 | f6 | 0e | 61 | 35 | 57 | b9 | 86 | c1 | 1d | 9e |
| e0 | e1 | f8 | 98 | 11 | 69 | d9 | 8e | 94 | 9b | 1e | 87 | e9 | ce | 55 | 28 | df |
| f0 | 8c | a1 | 89 | 0d | bf | e6 | 42 | 68 | 41 | 99 | 2d | 0f | b0 | 54 | bb | 16 |

# Shift Rows

Permute the bytes of State by applying a cyclic shift to each row.



This step is the most mathematically complicated step in the algorithm.




$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \cdot \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

# Mix Columns (as a bit operation)

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \cdot \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 2a_0 + 3a_1 + 1a_2 + 1a_3 \\ 1a_0 + 2a_1 + 3a_2 + 1a_3 \\ 1a_0 + 1a_1 + 2a_2 + 3a_3 \\ 3a_0 + 1a_1 + 1a_2 + 2a_3 \end{bmatrix}$$

- Each $a_i$ and $b_i$ is a *byte*. Regard bytes as 8-bit arrays.
- Each addition operation is a *bitwise* exclusive or.
- Multiplication by $1$ is the identity.
- Multiplication by $2$ is the following operation:
    - If the left-most bit is $0$, then perform a cyclic left shift (e.g. $01100111 \mapsto 11001110$)
    - If the left-most bit is $1$, then discard the $1$, insert a $0$ on the right, and XOR with `0x1b` $=$ `00011011` (e.g. $11001110 \mapsto 10011100 \oplus 00011011 = 10000111$)

# Mix Columns (as a matrix operation)

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \cdot \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 2a_0 + 3a_1 + 1a_2 + 1a_3 \\ 1a_0 + 2a_1 + 3a_2 + 1a_3 \\ 1a_0 + 1a_1 + 2a_2 + 3a_3 \\ 3a_0 + 1a_1 + 1a_2 + 2a_3 \end{bmatrix}$$

*Alternatively*, we can regard `MixColumns` as a matrix transformation in $\mathrm{GF}(2^8)$:

- Regard all bytes as polynomials in the finite field $\mathrm{GF}(2^8) = \mathbb{F}_2[x]/(x^8 + x^4 + x^3 + x + 1)$.
- Regard all integers in the matrix $(1, 2, 3)$ as bytes via their binary representations (e.g. $3 = 00000011 = x + 1$).
- Perform all additions and multiplications in the finite field $\mathrm{GF}(2^8)$.

# Mix Columns (as a polynomial operation)

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \cdot \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 2a_0 + 3a_1 + 1a_2 + 1a_3 \\ 1a_0 + 2a_1 + 3a_2 + 1a_3 \\ 1a_0 + 1a_1 + 2a_2 + 3a_3 \\ 3a_0 + 1a_1 + 1a_2 + 2a_3 \end{bmatrix}$$

A third alternative is to regard `MixColumns` as a polynomial multiplication.

- The `MixColumns` matrix is a *cyclic* matrix: each row is a rotation of the previous row.
- Multiplication by a cyclic $N \times N$ matrix corresponds to polynomial multiplication modulo $X^N - 1$.
- Regard the column of $a_i$'s as a polynomial in $\mathrm{GF}(2^8)[X]$: $a_0 + a_1 X + a_2 X^2 + a_3 X^3$
- Modulo $X^4 - 1$, we compute

$$(02 + 01 \cdot X + 01 \cdot X^2 + 03 \cdot X^3) \cdot (a_0 + a_1 X + a_2 X^2 + a_3 X^3)$$

to obtain $b_0 + b_1 X + b_2 X^2 + b_3 X^3$

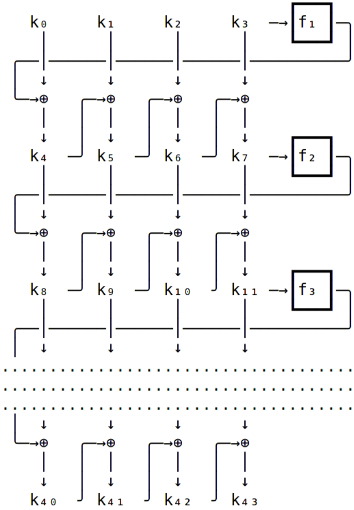
# AES Encryption

- From the key $k$, derive $h+1$ round keys $k_0, k_1, \ldots, k_h$ via the key schedule.
- The encryption function:

  State $\leftarrow$ plaintext
  for $i = 1 \ldots h-1$ do
  State $\leftarrow$ State $\oplus k_{i-1}$
  State $\leftarrow$ `SubBytes`(State)
  State $\leftarrow$ `ShiftRows`(State)
  State $\leftarrow$ `MixColumns`(State)
  State $\leftarrow$ State $\oplus k_{h-1}$
  State $\leftarrow$ `SubBytes`(State)
  State $\leftarrow$ `ShiftRows`(State)
  State $\leftarrow$ State $\oplus k_h$
  ciphertext $\leftarrow$ State

- Note that in the final round, `MixColumns` is not applied.

# AES key schedule (for 128-bit keys)




http://crypto.stackexchange.com/q/20

- For $128$-bit keys, AES has ten rounds, so we need eleven subkeys.
- Each $k_i$ is a 32-bit word (viewed as a 4-byte array).
- Each group of four $k_i$'s forms a $128$-bit subkey.
- The first round subkey $(k_0, k_1, k_2, k_3)$ equals the actual AES key.

# Key schedule core (for 128-bit keys)

The functions $f_i\colon \{0,1\}^{32} \to \{0,1\}^{32}$ are defined as follows:

- Left-shift the input cyclically by $8$ bits.
- Apply the AES S-box to each byte.
- Bitwise XOR the left-most byte with a constant which varies by round according to the following table.

| Round | constant | Round | constant |
|---|---|---|---|
| 1 | `0x01` | 6 | `0x20` |
| 2 | `0x02` | 7 | `0x40` |
| 3 | `0x04` | 8 | `0x80` |
| 4 | `0x08` | 9 | `0x1B` |
| 5 | `0x10` | 10 | `0x36` |

- Output the result.

# Outline

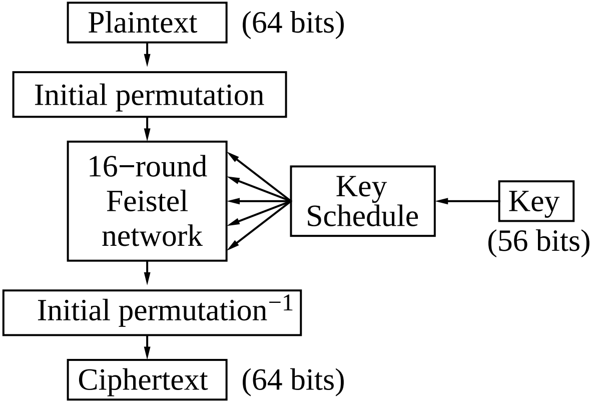
# Data Encryption Standard (DES)

- 1972: NBS (now NIST: National Institute of Standards and Technology) solicits proposals for encryption algorithms for the protection of computer data.
- 1974: IBM submits a variant of Lucifer (based on a Feistel network) as a DES candidate.
- 1975: NSA (National Security Agency) (allegedly) “fixes” DES
  - Reduces the key size from 64 bits to 56 bits.

  *“We sent the S-boxes off to Washington. They came back and were all different.”*
- 1977: DES adopted as US Federal Information Processing Standard (FIPS 46).
- 1981: DES adopted as a US banking standard (ANSI X3.92).

# Overview of DES

Block cipher with 64-bit blocks, 56-bit key, and 16 rounds of operation.



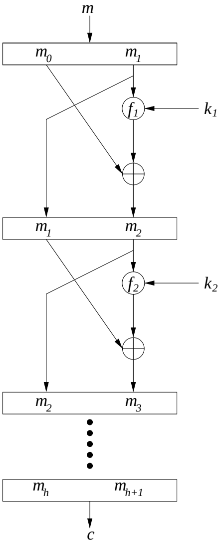
# Cryptanalysis of DES

*"DES did more to galvanize the field of cryptanalysis than anything else. Now there was an algorithm to study."* *—Bruce Schneier*

- Brute force attacks (try every key):
  - (1977 estimate) \$20 million machine to find keys in one day
  - (1993 estimate) \$1 million machine to find keys in 7 hours
  - (1999) EFF DES Cracker: \$250,000 machine, 4.5 days per key
  - (2006) COPACOBANA: \$10,000 machine, 4.5 days per key
  - (2012) Cloudcracker.com: \$200 and 11.5 hours per key
- Non-brute-force attacks:
  - Differential cryptanalysis (Eli Biham & Adi Shamir, 1991): $2^{49}$ chosen plaintexts
  - Linear cryptanalysis (Mitsuru Matsui, 1993): $2^{43}$ known plaintexts

# Outline

# The Feistel network design




- DES uses a Feistel network design.
- Plaintext is divided into two halves.
- Key is used to generate subkeys $k_1, k_2, \ldots, k_h$
- $f_i$ is a *component function* whose output value depends on $k_i$ and $m_i$

# Feistel Ciphers: A Class of Block Ciphers

Components of a Feistel cipher:

- Parameters: $n$ (half the block length), $h$ (number of rounds), $\ell$ (key size).
- $M = \{0,1\}^{2n}$, $C = \{0,1\}^{2n}$, $K = \{0,1\}^{\ell}$.
- A key scheduling algorithm which determines subkeys $k_1, k_2, \ldots, k_h$ from a key $k$.
- Each subkey $k_i$ defines a component function $f_i : \{0,1\}^{\ell} \times \{0,1\}^n \to \{0,1\}^n$.

# Components of a Feistel Cipher

Encryption takes $h$ rounds:

- Plaintext is $m = (m_0, m_1)$, where $m_i \in \{0,1\}^n$.
- Round 1: $(m_0, m_1) \mapsto (m_1, m_2)$, where $m_2 = m_0 \oplus f_1(k_1, m_1)$.
- Round 2: $(m_1, m_2) \mapsto (m_2, m_3)$, where $m_3 = m_1 \oplus f_2(k_2, m_2)$. $\quad\vdots$
- Round $h$: $(m_{h-1}, m_h) \mapsto (m_h, m_{h+1})$, where $m_{h+1} = m_{h-1} \oplus f_h(f_h, m_h)$.
- Ciphertext is $c = (m_h, m_{h+1})$.

Decryption: Given $c = (m_h, m_{h+1})$ and $k$, to find $m = (m_0, m_1)$:

- Compute $m_{h-1} = m_{h+1} \oplus f_h(k_h, m_h)$.
- Similarly, compute $m_{h-2}, \ldots, m_1, m_0$.

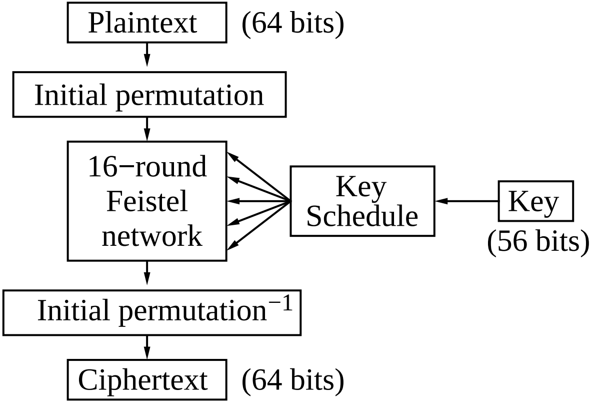
# Feistel Cipher (notes)

- No restrictions on the functions $f_i$ in order for the encryption procedure to be invertible.
- Underlying principle: Take something "simple" and use it several times; hope that the result is "complicated"
- Implementation:
  - Encryption: Only need to implement one round; the same code can be used for each round.
  - Decryption uses the same code as for encryption. (Use subkeys in reverse order.)

# Outline

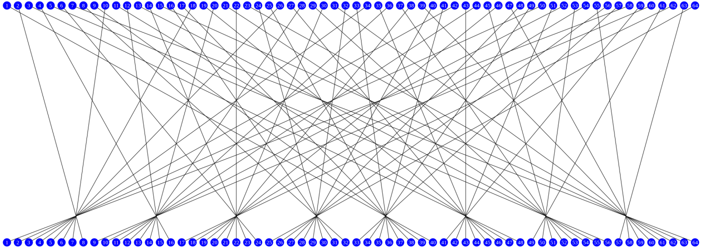
# Overview of DES

Feistel cipher with $n = 32$, $h = 16$, $\ell = 56$.



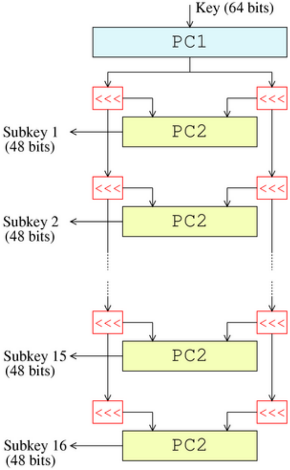
# Initial permutation

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| In | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| Out | 40 | 8 | 48 | 16 | 56 | 24 | 64 | 32 | 39 | 7 | 47 | 15 | 55 | 23 | 63 | 31 |
| In | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 |
| Out | 38 | 6 | 46 | 14 | 54 | 22 | 62 | 30 | 37 | 5 | 45 | 13 | 53 | 21 | 61 | 29 |
| In | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 |
| Out | 36 | 4 | 44 | 12 | 52 | 20 | 60 | 28 | 35 | 3 | 43 | 11 | 51 | 19 | 59 | 27 |
| In | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 |
| Out | 34 | 2 | 42 | 10 | 50 | 18 | 58 | 26 | 33 | 1 | 41 | 9 | 49 | 17 | 57 | 25 |



Credit: Wikipedia

# Key scheduling algorithm




| Round number | Left shift each half by this many bits |
|---|---|
| 1 | 1 |
| 2 | 1 |
| 3 | 2 |
| 4 | 2 |
| 5 | 2 |
| 6 | 2 |
| 7 | 2 |
| 8 | 2 |
| 9 | 1 |
| 10 | 2 |
| 11 | 2 |
| 12 | 2 |
| 13 | 2 |
| 14 | 2 |
| 15 | 2 |
| 16 | 1 |

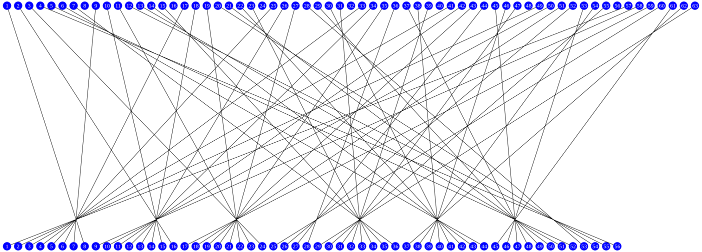
# Permuted Choice #1

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| In | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| Out | 8 | 16 | 24 | 56 | 52 | 44 | 36 | | 7 | 15 | 23 | 55 | 51 | 43 | 35 | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| In | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 |
| Out | 6 | 14 | 22 | 54 | 50 | 42 | 34 | | 5 | 13 | 21 | 53 | 49 | 41 | 33 | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| In | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 |
| Out | 4 | 12 | 20 | 28 | 48 | 40 | 32 | | 3 | 11 | 19 | 27 | 47 | 39 | 31 | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| In | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 |
| Out | 2 | 10 | 18 | 26 | 46 | 38 | 30 | | 1 | 9 | 17 | 25 | 45 | 37 | 29 | |



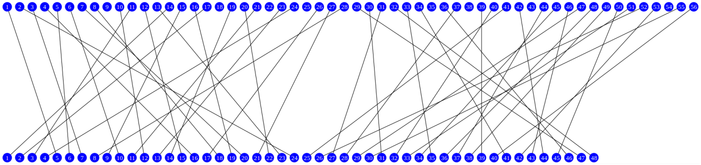
# Permuted Choice #2

| In | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Out | 5 | 24 | 7 | 16 | 6 | 10 | 20 | 18 | | 12 | 3 | 15 | 23 | 1 |

| In | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Out | 9 | 19 | 2 | | 14 | 22 | 11 | | 13 | 4 | | 17 | 21 | 8 |

| In | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Out | 47 | 31 | 27 | 48 | 35 | 41 | | 46 | 28 | | 39 | 32 | 25 | 44 |

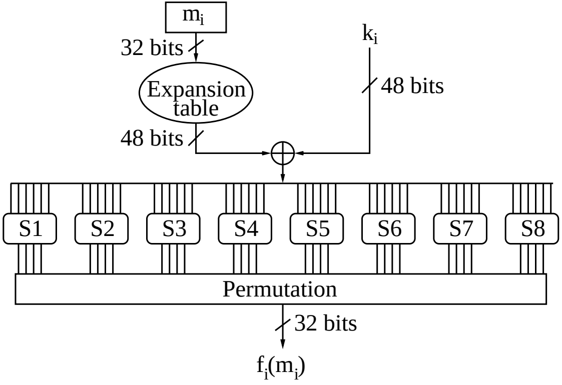
| In | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Out | | 37 | 34 | 43 | 29 | 36 | 38 | 45 | 33 | 26 | 42 | | 30 | 40 |



# Structure of the component functions

Recall $f_i : \{0,1\}^{32} \rightarrow \{0,1\}^{32}$.

Note: The IP, key scheduling algorithm, Expansion table, S-boxes, and Permutation are fixed and public knowledge.



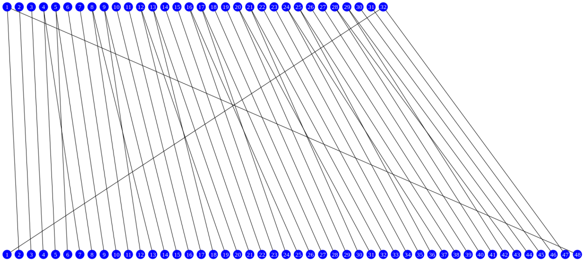
# Expansion table

| In | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| Out | 2, 48 | 3 | 4 | 5, 7 | 6, 8 | 9 | 10 | 11, 13 |

| In | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|
| Out | 12, 14 | 15 | 16 | 17, 19 | 18, 20 | 21 | 22 | 23, 25 |

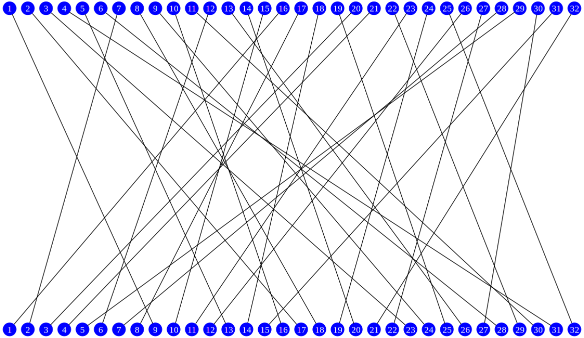
| In | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
|---|---|---|---|---|---|---|---|---|
| Out | 24, 26 | 27 | 28 | 29, 31 | 30, 32 | 33 | 34 | 35, 37 |

| In | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 |
|---|---|---|---|---|---|---|---|---|
| Out | 36, 38 | 39 | 40 | 41, 43 | 42, 44 | 45 | 46 | 1, 47 |



| In | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Out | 9 | 17 | 23 | 31 | 13 | 28 | 2 | 18 | 24 | 16 | 30 | 6 | 26 | 20 | 10 | 1 |

| In | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Out | 8 | 14 | 25 | 3 | 4 | 29 | 11 | 19 | 32 | 12 | 22 | 7 | 5 | 27 | 15 | 21 |



# DES S-boxes

Columns denote middle four bits of input. Rows denote outer two bits of input.

| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S_1$ | 0 | 14 | 4 | 13 | 1 | 2 | 15 | 11 | 8 | 3 | 10 | 6 | 12 | 5 | 9 | 0 | 7 |
| | 1 | 0 | 15 | 7 | 4 | 14 | 2 | 13 | 1 | 10 | 6 | 12 | 11 | 9 | 5 | 3 | 8 |
| | 2 | 4 | 1 | 14 | 8 | 13 | 6 | 2 | 11 | 15 | 12 | 9 | 7 | 3 | 10 | 5 | 0 |
| | 3 | 15 | 12 | 8 | 2 | 4 | 9 | 1 | 7 | 5 | 11 | 3 | 14 | 10 | 0 | 6 | 13 |
| $S_2$ | 0 | 15 | 1 | 8 | 14 | 6 | 11 | 3 | 4 | 9 | 7 | 2 | 13 | 12 | 0 | 5 | 10 |
| | 1 | 3 | 13 | 4 | 7 | 15 | 2 | 8 | 14 | 12 | 0 | 1 | 10 | 6 | 9 | 11 | 5 |
| | 2 | 0 | 14 | 7 | 11 | 10 | 4 | 13 | 1 | 5 | 8 | 12 | 6 | 9 | 3 | 2 | 15 |
| | 3 | 13 | 8 | 10 | 1 | 3 | 15 | 4 | 2 | 11 | 6 | 7 | 12 | 0 | 5 | 14 | 9 |
| $S_3$ | 0 | 10 | 0 | 9 | 14 | 6 | 3 | 15 | 5 | 1 | 13 | 12 | 7 | 11 | 4 | 2 | 8 |
| | 1 | 13 | 7 | 0 | 9 | 3 | 4 | 6 | 10 | 2 | 8 | 5 | 14 | 12 | 11 | 15 | 1 |
| | 2 | 13 | 6 | 4 | 9 | 8 | 15 | 3 | 0 | 11 | 1 | 2 | 12 | 5 | 10 | 14 | 7 |
| | 3 | 1 | 10 | 13 | 0 | 6 | 9 | 8 | 7 | 4 | 15 | 14 | 3 | 11 | 5 | 2 | 12 |
| $S_4$ | 0 | 7 | 13 | 14 | 3 | 0 | 6 | 9 | 10 | 1 | 2 | 8 | 5 | 11 | 12 | 4 | 15 |
| | 1 | 13 | 8 | 11 | 5 | 6 | 15 | 0 | 3 | 4 | 7 | 2 | 12 | 1 | 10 | 14 | 9 |
| | 2 | 10 | 6 | 9 | 0 | 12 | 11 | 7 | 13 | 15 | 1 | 3 | 14 | 5 | 2 | 8 | 4 |
| | 3 | 3 | 15 | 0 | 6 | 10 | 1 | 13 | 8 | 9 | 4 | 5 | 11 | 12 | 7 | 2 | 14 |
| $S_5$ | 0 | 2 | 12 | 4 | 1 | 7 | 10 | 11 | 6 | 8 | 5 | 3 | 15 | 13 | 0 | 14 | 9 |
| | 1 | 14 | 11 | 2 | 12 | 4 | 7 | 13 | 1 | 5 | 0 | 15 | 10 | 3 | 9 | 8 | 6 |
| | 2 | 4 | 2 | 1 | 11 | 10 | 13 | 7 | 8 | 15 | 9 | 12 | 5 | 6 | 3 | 0 | 14 |
| | 3 | 11 | 8 | 12 | 7 | 1 | 14 | 2 | 13 | 6 | 15 | 0 | 9 | 10 | 4 | 5 | 3 |
| $S_6$ | 0 | 12 | 1 | 10 | 15 | 9 | 2 | 6 | 8 | 0 | 13 | 3 | 4 | 14 | 7 | 5 | 11 |
| | 1 | 10 | 15 | 4 | 2 | 7 | 12 | 9 | 5 | 6 | 1 | 13 | 14 | 0 | 11 | 3 | 8 |
| | 2 | 9 | 14 | 15 | 5 | 2 | 8 | 12 | 3 | 7 | 0 | 4 | 10 | 1 | 13 | 11 | 6 |
| | 3 | 4 | 3 | 2 | 12 | 9 | 5 | 15 | 10 | 11 | 14 | 1 | 7 | 6 | 0 | 8 | 13 |
| $S_7$ | 0 | 4 | 11 | 2 | 14 | 15 | 0 | 8 | 13 | 3 | 12 | 9 | 7 | 5 | 10 | 6 | 1 |
| | 1 | 13 | 0 | 11 | 7 | 4 | 9 | 1 | 10 | 14 | 3 | 5 | 12 | 2 | 15 | 8 | 6 |
| | 2 | 1 | 4 | 11 | 13 | 12 | 3 | 7 | 14 | 10 | 15 | 6 | 8 | 0 | 5 | 9 | 2 |
| | 3 | 6 | 11 | 13 | 8 | 1 | 4 | 10 | 7 | 9 | 5 | 0 | 15 | 14 | 2 | 3 | 12 |
| $S_8$ | 0 | 13 | 2 | 8 | 4 | 6 | 15 | 11 | 1 | 10 | 9 | 3 | 14 | 5 | 0 | 12 | 7 |
| | 1 | 1 | 15 | 13 | 8 | 10 | 3 | 7 | 4 | 12 | 5 | 6 | 11 | 0 | 14 | 9 | 2 |
| | 2 | 7 | 11 | 4 | 1 | 9 | 12 | 14 | 2 | 0 | 6 | 10 | 13 | 15 | 3 | 5 | 8 |
| | 3 | 2 | 1 | 14 | 7 | 4 | 10 | 8 | 13 | 15 | 12 | 9 | 0 | 3 | 5 | 6 | 11 |

# DES S-boxes

Substitution-boxes or S-boxes ($S_1$, $S_2$, $S_3$, $S_4$, $S_5$, $S_6$, $S_7$, $S_8$):

- Each S-box is a function taking six input bits and producing four output bits.
- S-boxes are the only components of DES that are non-linear. (Without the S-boxes, changing one plaintext bit would change very few ciphertext bits.)
- Security of DES crucially depends on their choice.
- DES with randomly selected S-boxes is easy to break.

# Performance

Speed benchmarks for software implementation on an Intel Core i9 2.9 GHz six-core Coffee Lake (8950HK) using OpenSSL 1.1.1c

| SKES | Block length (bits) | Key length (bits) | Speed (Mbytes/sec) |
|---|---|---|---|
| RC4 | — | 128 | 647 |
| ChaCha20 | — | 256 | 1497 |
| DES | 64 | 56 | 101 |
| 3DES | 64 | (112) | 38 |
| AES (software) | 128 | 128 | 185 |
| AES (AES-NI) | 128 | 128 | 1647 |

# Outline

# DES Problem 1: Small Key Size

- Exhaustive search on key space takes $2^{56}$ steps and can be easily parallelized.

# Quick response question

Suppose you have one million ($2^{20}$) computers, each of which runs at 2 GHz (i.e., $2^{31}$) cycles per second. Suppose it takes $2^{15}$ cycles to do one block of DES encryption. How long would it take to do an exhaustive key search on DES's 56-bit keys?

# DES Problem 1: Small Key Size

- Exhaustive search on key space takes $2^{56}$ steps and can be easily parallelized.
- DES challenges from RSA Security (3 known PT/CT pairs):

| | | | |
|---|---|---|---|
| The unkn | own mess | age is: | ???????? |

  - June 1997: Broken by Internet search (3 months).
  - July 1998: Broken in 3 days by DeepCrack machine (1800 chips; $250,000).
  - Jan 1999: Broken in 22 hrs, 15 min (DeepCrack + `http://distributed.net`).

# DES Problem 2: Small Block Size

- If plaintext blocks are distributed "uniformly at random", then the expected number of ciphertext blocks observed before a collision occurs is $\approx 2^{32}$ (by the birthday paradox).
    - Hence the ciphertext reveals some information about the plaintext.
- Small block length is also damaging to some authentication applications (more on this later).

# Sophisticated Attacks on DES

Differential cryptanalysis [Biham & Shamir 1989]:

- Recovers key given $2^{47}$ chosen plaintext/ciphertext pairs.
- DES was designed to resist this attack.
- Differential cryptanalysis has been more effective on some other block ciphers.

Linear cryptanalysis [Matsui 1993]:

- Recovers key given $2^{43}$ known plaintext/ciphertext pairs.
- Storing these pairs takes 131,000 Gbytes.
- Implemented in 1993: 10 days on 12 machines.

# Outline

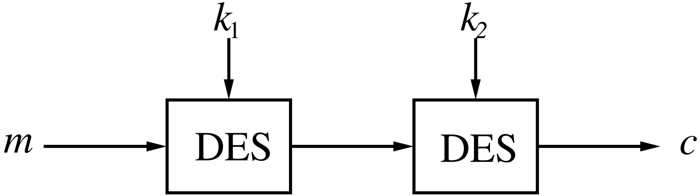
# Mitigating short keys: encrypt multiple times

- Multiple encryption: Re-encrypt the ciphertext one or more times using independent keys, and hope that this operation increases the effective key length.
- Multiple encryption does not always increase security.
  - Example: If $E_\pi$ denotes the simple substitution cipher with key $\pi$, then is $E_{\pi_1} \circ E_{\pi_2}$ any more secure than $E_\pi$?

# Double encryption

- Double-DES. Key is $k = (k_1, k_2)$, $k_1, k_2 \in_R \{0,1\}^{56}$.
- Encryption: $c = E_{k_2}(E_{k_1}(m))$.

  ($E$ = DES encryption, $E^{-1}$ = DES decryption)




- Decryption: $m = E^{-1}_{k_1}(E^{-1}_{k_2}(c))$.
- Key size of Double-DES is $\ell = 112$, so exhaustive key search takes $2^{112}$ steps (infeasible).
- Note: Block length is unchanged.

# Attack on Double-DES

Main idea: If $c = E_{k_2}(E_{k_1}(m))$, then $E_{k_2}^{-1}(c) = E_{k_1}(m)$. (Meet-in-the-middle)

1. Given: Known plaintext pairs $(m_i, c_i)$, $i = 1, 2, 3, \ldots$
2. For each $h_2 \in \{0,1\}^{56}$:
   2.1 Compute $E_{h_2}^{-1}(c_1)$, and store $[E_{h_2}^{-1}(c_1), h_2]$ in a table.
3. For each $h_1 \in \{0,1\}^{56}$ do the following:
   3.1 Compute $E_{h_1}(m_1)$.
   3.2 Search for $E_{h_1}(m_1)$ in the table.
   3.3 If $E_{h_1}(m_1) = E_{h_2}^{-1}(c_1)$:
      - Check if $E_{h_1}(m_2) = E_{h_2}^{-1}(c_2)$
      - Check if $E_{h_1}(m_3) = E_{h_2}^{-1}(c_3)$

      $$\vdots$$

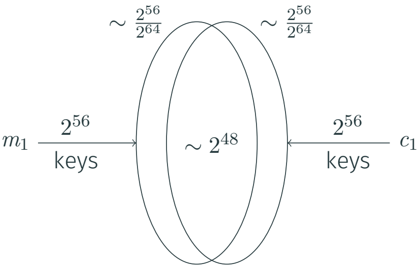
      If all checks pass, then output $(h_1, h_2)$ and STOP.

Complexity of the attack is $\approx 2^{57}$.

# Analyzing the meet-in-the-middle attack

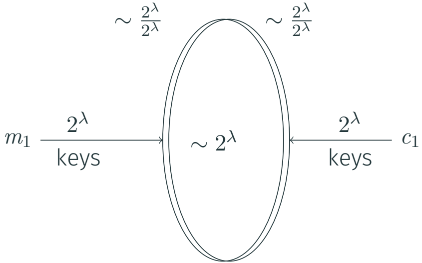
- Number of known plaintext/ciphertext pairs required to avoid false keys: 2 suffice, with high probability
- Number of DES operations is $\approx 2^{56} + 2^{56} + 2 \cdot 2^{48} \approx 2^{57}$.
    - We are not counting the time to do the sorting and searching
- Space requirements: $2^{56}(64 + 56)$ bits $\approx 1,080,863$ Tbytes.

# More details on attack cost




- Approximately $2^{48}$ keys $h_1$ encrypting $m_1$ yield intermediate ciphertext equal to a decryption of $c_1$ under some key $h_2$.
- But with high probability $E_{h_1}(m_2) \neq E_{h_2}(c_2)$
- So we have to do $\sim 2^{48}$ DES operations for checking next pair
- In total $2^{56} + 2^{56} + \sim 2^{48}$ DES operations

# Meet-in-the-middle attack with key size = block size = $\lambda$




- Approximately $2^\lambda$ keys $h_1$ encrypting $m_1$ yield intermediate ciphertext equal to a decryption of $c_1$ under some key $h_2$.
- But with high probability $E_{h_1}(m_2) \neq E_{h_2}(c_2)$
- So we have to do $\sim 2^\lambda$ DES operations for checking next pair
- In total $2^\lambda + 2^\lambda + \sim 2^\lambda$ DES operations

# Analyzing the meet-in-the-middle attack

Time-memory tradeoff. [Exercise] The attack can be modified to decrease the storage requirements at the expense of time:

- Time: $2^{56+s}$ steps; memory: $2^{56-s}$ units, $1 \leq s \leq 55$.

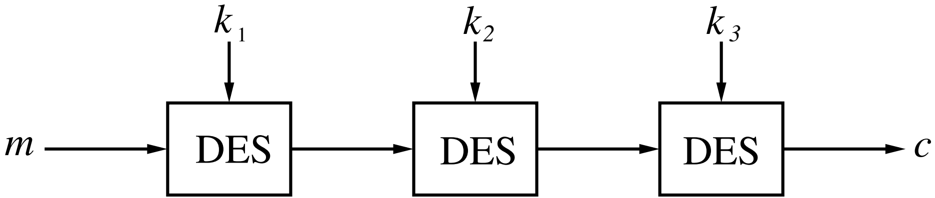
Conclusions:

- Double-DES has the same effective key length as DES.
- Double-DES is not much more secure than DES.

# Three-key Triple encryption

- Triple-DES. Key is $k = (k_1, k_2, k_3)$, $k_1, k_2, k_3 \in_R \{0,1\}^{56}$.
- Encryption: $c = E_{k_3}(E_{k_2}(E_{k_1}(m)))$.

  ($E$ = DES encryption, $E^{-1}$ = DES decryption)



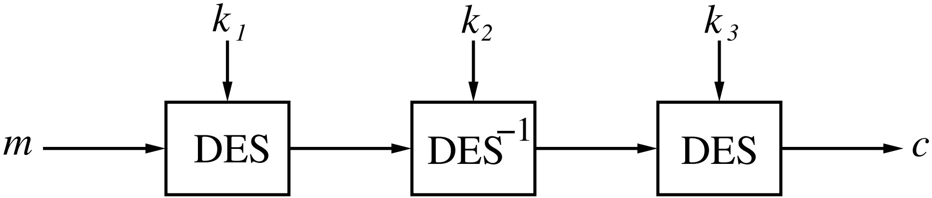

- Decryption: $m = E_{k_1}^{-1}(E_{k_2}^{-1}(E_{k_3}^{-1}(c)))$.
- Key length of Triple-DES is $\ell = 168$, so exhaustive key search takes $2^{168}$ steps (infeasible).

# Three-key Triple DES

- Meet-in-the-middle attack takes $\approx 2^{112}$ steps. [Exercise]
- So, the effective key length of Triple-DES against exhaustive key search is $\leq$ 112 bits.
- No proof that Triple-DES is more secure than DES.
- Note: Block length is 64 bits, and now forms the weak link:
  - Adversary stores a large table (of size $\leq 2^{64}$) of $(m, c)$ pairs (dictionary attack).
  - To prevent this attack: change secret keys frequently.
- Triple-DES is widely deployed.

# Some variants

EDE Triple-DES: for backward compatibility with DES




Two-key Triple-DES: