

Assume encryption and decryption use the same key.  
Will discuss how to distribute key to all parties later  
Symmetric ciphers unusable for authentication of sender

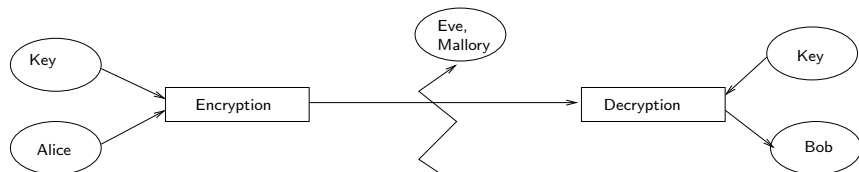
Kinds of symmetric ciphers:

- **Block cipher**: Symmetric cipher operating on fixed-length groups of bits, called blocks
- **Stream cipher** Symmetric cipher encrypting plaintext continuously. Bits are encrypted one at a time, differently for each bit.

# Players

Have the following main players:

- **Alice**: sender of an encrypted message
- **Bob**: intended receiver of encrypted message. Assumed to the key.
- **Eve**: (Passive) attacker intercepting messages and trying to identify plaintexts or keys
- **Mallory**: (Active) attacker intercepting and modifying messages to identify plaintexts or keys



# Feistel cipher: a way of doing block ciphers

Invented in 1971 at IBM

Important class of ciphers (eg Blowfish, DES, 3DES)

Same encryption scheme applied iteratively for several rounds

Important step: Derive next message state from previous message state via special function called *Feistel function*

Encryption is organised as a series of “rounds”.

Each round works as follows:

- Split input in half
- Apply Feistel function to the right half
- Compute xor of result with old left half to be new left half
- Swap old right and new left half, unless we are in the last round

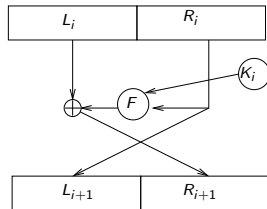
## Feistel Cipher, continued

Formal definition:

- Split plaintext block in two equal pieces  $M = (L_0, R_0)$
- For each round  $i = 0, 1, \dots, r - 1$  compute

$$\begin{aligned}L_{i+1} &= R_i \\R_{i+1} &= L_i \oplus F(K_i, R_i)\end{aligned}$$

- The ciphertext is  $C = (R_r, L_r)$



# Decryption

Works as encryption, but with a reversed order of keys

- Split ciphertext block in two equal pieces  $C = (R_r, L_r)$
- For each round  $i = r, r - 1, \dots, 1$  compute

$$\begin{aligned}R_{i-1} &= L_i \\L_{i-1} &= R_i \oplus F(K_{i-1}, L_i)\end{aligned}$$

- Plaintext is  $M = (L_0, R_0)$

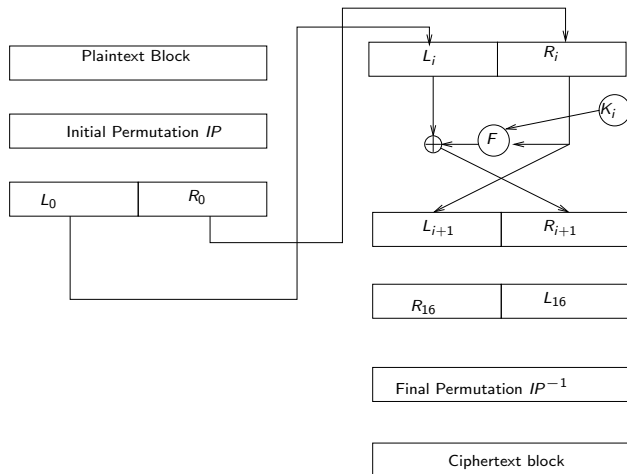
# DES

Data Encryption Standard (DES) adopted in 1976

Key size (56 bits) is too small for today's computers (can be broken within 10 hours)

Variants still provide good security

# Overview of DES



# Design parameters

- Block length is 64 bits
- Number of rounds  $R$  is 16
- Key length is 56 bits
- Round key length is 48 bit for each subkey  $K_0, \dots, K_{15}$ .  
Subkeys are derived from 56 bit key via special key schedule.

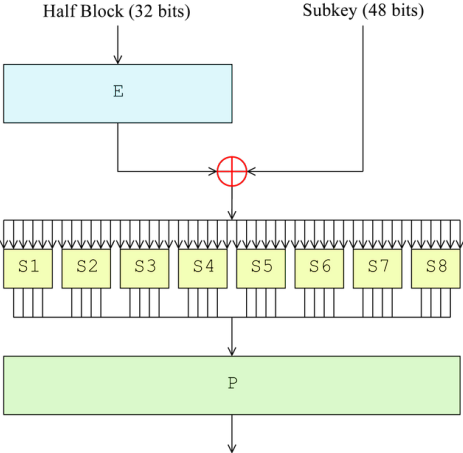


# DES Feistel function

Four stage procedure:

- **Expansion permutation:** Expand 32-bit message half block to 48 bit block by doubling 16 bits and permuting them
- **Round key addition:** Compute xor of this 48 bit block with round key  $K_i$
- **S-Box:** Split 48 bit into eight 6-bit blocks. Each of them is given as input to eight substitution boxes, which substitute 6-bit block by 4-bit block.
- **P-Box:** Combine these eight 4-bit blocks to 32-bit block and apply another permutation.

# DES Feistel function, continued



Source: Wikipedia

# Notation for DES operations

Have three special operations:

- **Cyclic shifts** on bitstring blocks: Will denote by  $b \lll n$  the move of the bits of block  $b$  by  $n$  to the left. Bits that would have fallen out are added at the right side of the  $b$ .  $b \ggg n$  is defined similarly
- **Permutations on the position of bits**: Written down as output order of the input bits.

Example: the permutation 

4	1	2	3
---	---	---	---

 means that

- the fourth input bit becomes the first output bit,
- the first input bit becomes the second output bit,
- the second input bit becomes the third output bit, and
- the third input bit becomes the fourth output bit.

Sometimes, we use the word “permutation” for bit re-arrangements that include duplication or dropping of bits, even though that is not a proper permutation.

# S-boxes

- **S-boxes**: An S-box substitution is a table lookup. Input is 6 bit, output is 4 bit. Works as follows:
  - Strip out outer bits of input and join them. This two-bit number is the row index.
  - Four inner bits indicate column number.
  - Output is corresponding entry in table

## Key schedule

Have different keys for each round, computed by so-called *Key schedule*

64-bit key is actually 56-bit key plus 8 parity bits

- First apply a permutation PC-1 which removes the parity bits. This results in 56 bits.
- Split result into half to obtain  $(C_0, D_0)$
- For each round we compute

$$\begin{aligned}C_i &= C_{i-1} \lll p_i \\D_i &= D_{i-1} \lll p_i\end{aligned}$$

where

$$p_i = \begin{cases} 1 & \text{if } i = 1, 2, 9, 16 \\ 2 & \text{otherwise} \end{cases}$$

- Now we join  $C_i$  and  $D_i$  together, and apply a permutation PC-2 which produces a 48-bit output.

## Security of block ciphers

To define the security of block ciphers, we look at a more abstract notion: pseudorandom permutations.

### Definition

Let  $X = \{0, 1\}^n$ . A *pseudorandom permutation* over  $(K, X)$  is a function

$$E: K \times X \rightarrow X$$

such that

- ▶ there exists an efficient deterministic algorithm to compute  $E(k, x)$  for any  $k$  and  $x$ ;
- ▶ The function  $E(k, \_)$  is one-to-one for each  $k$
- ▶ There exists a function  $D: K \times X \rightarrow X$  which is efficiently computable, and  $D(k, E(k, x)) = x$  for all  $k$  and  $x$ .

# Security of pseudorandom permutations

A pseudorandom permutation is secure if an adversary (who can call it) can't distinguish it from a "genuine" random permutation. Suppose  $X$  and  $K$  have size  $N$ , i.e.,  $X = K = \{0, 1\}^n$ .

- ▶ There are  $N! = 2^n!$  permutations  $X \rightarrow X$ .
- ▶ There are  $|K| = 2^n$  pseudorandom permutations.

For example, suppose  $n=64$ . Then these numbers are, very roughly,  $(10^{19})^{10^{19}}$  and  $10^{19}$ .

So there are much fewer pseudorandom permutations there are permutations in total.

## Definition

Let  $X = \{0, 1\}^n$ , and  $\mathcal{F}$  be the set of all permutations on  $X$ , and  $E$  a pseudorandom permutation over  $(K, X)$ . Define the following game between the attacker and the challenger:

- ▶ The challenger chooses a random bit  $b \in \{0, 1\}$ .
- ▶ If  $b = 0$ , the challenger chooses a  $k \in K$  at random, and if  $b = 1$ , the challenger chooses a permutation  $f$  on  $X$  at random.
- ▶ The attacker does arbitrary computations.
- ▶ The attacker has access to a black box, which is a function from  $X$  to  $X$  operated by the challenger. He can ask the challenger for the values  $g(x_1), \dots, g(x_n)$  during his computation.
- ▶ If  $b = 0$ , the challenger answers the query  $g(x_i)$  by returning  $E(k, x_i)$ , and if  $b = 1$ , the answer is  $f(x_i)$ .
- ▶ Eventually the attacker outputs a bit  $b' \in \{0, 1\}$ .

The attacker wins this game if  $b = b'$ .



# The attacker's power in security games

In security games, attacker can only do efficient operations, and only “efficiently” many of them

Formally: attacker is *probabilistic polynomial-time Turing machine* (PPT)

Importantly: attacker cannot search through all keys, as the number of possible keys increases exponentially with the length of the key

## Definition

A function  $\epsilon : \mathbb{N} \rightarrow \mathbb{R}^+$  is called *negligible* if for all  $d$  there exists a  $x_d$  such that for all  $x \geq x_d$ ,

$$\epsilon(x) \leq \frac{1}{x^d}$$

## Definition

A pseudorandom permutation  $E : K \times X \rightarrow X$  is *secure* if for all PPT attackers  $A$ ,

$$\left| \Pr[b = b'] - \frac{1}{2} \right|$$

is negligible in the size of  $K$ .

Note that  $\left| \Pr[b = b'] - \frac{1}{2} \right|$  is a function of the size of  $K$ .

## Example

1. Let  $X = \{0, 1\}^n$  and  $K = \{1, \dots, n\}$ .

Let  $E(k, x)$  be computed as follows:

Apply the Rail Fence cipher bitwise to  $x$  with key  $k$ .

Is that a secure pseudorandom permutation?

## Example

- Let  $X = \{A, B, \dots, Z\}^n$  and  $K = \{\text{the set of permutations on } \{A, B, \dots, Z\}\}$ .

Let  $E(k, x)$  be computed as follows: apply the permutation  $k$  to each of the characters  $x$  in turn.

Is that a secure pseudorandom permutation?

# What we know about DES

DES a good design, but as it only has 56 bit keys, it has only approximately  $2^{56}$  security. (There are some cryptanalytic attacks on DES, but not very serious ones, so let's say its security is about  $2^{56}$ .)

How about using DES twice? Take a 112-bit key, split it into two keys  $K_1$  and  $K_2$  and encrypt  $M$  like this:

$$\text{Enc}_{K_1}(\text{Enc}_{K_2}(M))$$

Would that give us  $2^{112}$  security?

## “2DES” is not significantly more secure than DES

Suppose we have a pair  $(M, C)$  consisting of a valid plaintext-ciphertext pair. With approximately  $2^{57}$  work, we can find the 112-bit key  $K_1K_2$  used in 2DES. Here is how to do it.

- ▶ Try all  $2^{56}$  possible keys  $K_2$ , and store all the results  $\text{Enc}_{K_2}(M)$ . Sort them in order. This is  $2^{56}$  work for the encryption, and  $2^{56} \log(2^{56})$  for the sorting.
- ▶ Try all the  $2^{56}$  possible keys  $K_1$ , computing  $\text{Dec}_{K_1}(C)$ . For each such value, check if it is one of the stored  $\text{Enc}_{K_2}(M)$ . That is  $2^{56}$  work for the Dec, and  $\log(2^{56})$  work for the checking.

The total work is not much more than  $2^{57}$ .

## 3DES is good, but slow

3DES takes the same idea, but uses DES three times. That gives us a 168-bit key. Take the 168-bit key, split it into three keys  $K_1$ ,  $K_2$  and  $K_3$ , and encrypt  $M$  like this:

$$\text{Enc}_{K_1}(\text{Dec}_{K_2}(\text{Enc}_{K_3}(M)))$$

- ▶ Why Enc-Dec-Enc instead of Enc-Enc-Enc?  
Enc-Dec-Enc gives us an option of setting  $K_1 = K_2 = K_3$ , which is then equivalent to DES. So if you have 3DES, you can make it do DES. This could be useful in some circumstances.
- ▶ How much security does 3DES give us? It doesn't give us  $2^{168}$  of security, because the same meet-in-the-middle attack as we had for "2DES" is possible. It is said to give us  $2^{118}$  of security.

## 3DES is good, but slow

3DES takes the same idea, but uses DES three times. That gives us a 168-bit key. Take the 168-bit key, split it into three keys  $K_1$ ,  $K_2$  and  $K_3$ , and encrypt  $M$  like this:

$$\text{Enc}_{K_1}(\text{Dec}_{K_2}(\text{Enc}_{K_3}(M)))$$

- ▶ Why Enc-Dec-Enc instead of Enc-Enc-Enc?  
Enc-Dec-Enc gives us an option of setting  $K_1 = K_2 = K_3$ , which is then equivalent to DES. So if you have 3DES, you can make it do DES. This could be useful in some circumstances.
- ▶ How much security does 3DES give us? It doesn't give us  $2^{168}$  of security, because the same meet-in-the-middle attack as we had for "2DES" is possible. It is said to give us  $2^{118}$  of security.