# CSE107: Intro to Modern Cryptography

https://cseweb.ucsd.edu/classes/sp22/cse107-a/

Emmanuel Thomé

April 5, 2022

# Lecture 3

## Block ciphers and Key-recovery security

Recall from last lecture

Notations, definitions

Definition of a block cipher

The DES block cipher

Two examples of formal attack scenarios

# Plan

Recall from last lecture

Notations, definitions

Definition of a block cipher

The DES block cipher

Two examples of formal attack scenarios

# Recall from last lecture: Perfect security

**Definition: perfect security**

Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme. We say that $\mathcal{SE}$ is perfectly secure if for any two messages $M_1, M_2 \in \text{Plaintexts}$ and any $C$

$$\Pr\left[\mathcal{E}_K(M_1) = C\right] = \Pr\left[\mathcal{E}_K(M_2) = C\right] .$$

In both cases, the probability is over the random choice $K \xleftarrow{\$} \mathcal{K}$ and over the coins tossed by $\mathcal{E}$ if any.

Intuitively: Given $C$, and even knowing the message is either $M_1$ or $M_2$ the adversary cannot determine which.

# Intuition for One-Time-Pad (OTP) security

Recall that One-Time-Pad encrypts $M$ to $\mathcal{E}_K(M) = K \oplus M$.

Suppose adversary gets ciphertext $C = 101$ and knows the plaintext $M$ is either $M_1 = 010$ or $M_2 = 001$. Can it tell which?

No, because $C = K \oplus M$ so

- $M = 010$ iff $K = 111$
- $M = 001$ iff $K = 100$

but $K$ is equally likely to be 111 or 100 and adversary does not know $K$.

# Perfect security of OTP

**Claim: OTP is perfectly secure**

Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be the OTP scheme with key-length $m \geq 1$. Then $\mathcal{SE}$ is perfectly secure.

Want to show that for any $M_1, M_2, C$

$$\Pr\left[\mathcal{E}_K(M_1) = C\right] = \Pr\left[\mathcal{E}_K(M_2) = C\right]$$

That is

$$\Pr\left[K \oplus M_1 = C\right] = \Pr\left[K \oplus M_2 = C\right]$$

when $K \xleftarrow{\$} \{0,1\}^m$.

# Example: $m = 2$

Messages:

| Keys: | 00 | 01 | 10 | 11 |
|---|---|---|---|---|
| 00 | 00 | 01 | 10 | 11 |
| 01 | 01 | 00 | 11 | 10 |
| 10 | 10 | 11 | 00 | 01 |
| 11 | 11 | 10 | 01 | 00 |

The entry in row $K$, column $M$ of the table is $\mathcal{E}_K(M) = K \oplus M$.

- $\Pr[\mathcal{E}_K(00) = 01] =$

# Example: $m = 2$

Messages:

| Keys: | | 00 | 01 | 10 | 11 |
|---|---|---|---|---|---|
| | 00 | 00 | 01 | 10 | 11 |
| | 01 | 01 | 00 | 11 | 10 |
| | 10 | 10 | 11 | 00 | 01 |
| | 11 | 11 | 10 | 01 | 00 |

The entry in row $K$, column $M$ of the table is $\mathcal{E}_K(M) = K \oplus M$.

- $\Pr[\mathcal{E}_K(00) = 01] = \frac{1}{4}$
- $\Pr[\mathcal{E}_K(10) = 01] =$

# Example: $m = 2$

Messages:

| Keys: | | 00 | 01 | 10 | 11 |
|---|---|---|---|---|---|
| | 00 | 00 | 01 | 10 | 11 |
| | 01 | 01 | 00 | 11 | 10 |
| | 10 | 10 | 11 | 00 | 01 |
| | 11 | 11 | 10 | 01 | 00 |

The entry in row $K$, column $M$ of the table is $\mathcal{E}_K(M) = K \oplus M$.

- $\Pr[\mathcal{E}_K(00) = 01] = \frac{1}{4}$
- $\Pr[\mathcal{E}_K(10) = 01] = \frac{1}{4}$

# Proof of claim

Probability for $M_1$

$$\Pr\left[\mathcal{E}_K(M_1) = C\right] \;=\; \Pr\left[K \oplus M_1 = C\right]$$

# Proof of claim

Probability for $M_1$

$$\begin{aligned}\Pr\left[\mathcal{E}_K(M_1) = C\right] &= \Pr\left[K \oplus M_1 = C\right] \\ &= \frac{|\{\, K \in \{0,1\}^m \,:\, K \oplus M_1 = C \,\}|}{|\{0,1\}^m|}\end{aligned}$$

# Proof of claim

Probability for $M_1$

$$
\begin{aligned}
\Pr\left[\mathcal{E}_K(M_1) = C\right] &= \Pr\left[K \oplus M_1 = C\right] \\
&= \frac{|\{\, K \in \{0,1\}^m \,:\, K \oplus M_1 = C \,\}|}{|\{0,1\}^m|} \\
&= \frac{1}{2^m}\,.
\end{aligned}
$$

# Proof of claim

Same for $M_2$

$$\begin{aligned}\Pr\left[\mathcal{E}_K(M_2) = C\right] &= \Pr\left[K \oplus M_2 = C\right] \\ &= \frac{|\{\, K \in \{0,1\}^m \,:\, K \oplus M_2 = C \,\}|}{|\{0,1\}^m|} \\ &= \frac{1}{2^m}\,.\end{aligned}$$

In fact, OTP is the only encryption scheme that achieves Shannon's perfect security.

# Perfect security: Plusses and Minuses

| + | - |
| --- | --- |
| Very good privacy | Key needs to be as long as message |

# What next

We want schemes to securely encrypt

- arbitrary amounts of data
- with a single, short (e.g., 128 bit) key

This will be possible once we relax our goal from perfect to computational security.

Plan:

- Study the primitives we will use, namely block ciphers
- Understand and define computational security of block ciphers and encryption schemes
- Use (computationally secure) block ciphers to build (computationally secure) encryption schemes

# Plan

# Notation

$\{0,1\}^n$ is the set of $n$-bit strings and $\{0,1\}^*$ is the set of all strings of finite length. By $\varepsilon$ we denote the empty string.

If $S$ is a set then $|S|$ denotes its size. Example: $|\{0,1\}^2| = 4$.

If $x$ is a string then $|x|$ denotes its length. Example: $|0100| = 4$.

If $m \geq 1$ is an integer then let $\mathbb{Z}_m = \{0, 1, \ldots, m-1\}$.

By $x \xleftarrow{\$} S$ we denote picking an element at random from set $S$ and assigning it to $x$. Thus

$$\Pr[x = s] = 1/|S| \text{ for every } s \in S.$$

# Functions

Let $n \geq 1$ be an integer. Let $X_1, \ldots, X_n$ and $Y$ be (non-empty) sets.

By $f\colon X_1 \times \cdots \times X_n \to Y$ we denote that $f$ is a function that

- Takes inputs $x_1, \ldots, x_n$, where $x_i \in X_i$ for $1 \leq i \leq n$
- and returns an output $y = f(x_1, \ldots, x_n) \in Y$.

We call $n$ the number of inputs (or arguments) of $f$. We call $X_1 \times \cdots \times X_n$ the domain of $f$ and $Y$ the range of $f$.

**Long notation:**

$$f : \begin{cases} X_1 \times \cdots \times X_n & \rightarrow & Y \\ (x_1, \ldots, x_n) & \mapsto & \text{some expression} \end{cases}$$

is a way to denote a function with domain $X_1 \times \cdots \times X_n$ and range $Y$, together with the mathematical expression that computes it.

# Example

**Example:** Define $f : \mathbb{Z}_3 \times \mathbb{Z}_3 \to \mathbb{Z}_3$ by $f(x_1, x_2) = (x_1 + x_2) \bmod 3$.
We can also write:

$$f : \begin{cases} \mathbb{Z}_3 \times \mathbb{Z}_3 & \to & \mathbb{Z}_3 \\ (x_1, x_2) & \mapsto & (x_1 + x_2) \bmod 3 \end{cases}$$

$f$ is a function with $n = 2$ inputs, domain $\mathbb{Z}_3 \times \mathbb{Z}_3$ and range $\mathbb{Z}_3$.

# Permutations

**Definition: permutation**

Suppose $f : X \to Y$ is a function with one argument. We say that it is a *permutation* if

- $X = Y$, meaning its domain and range are the same set.
- There is an *inverse* function $f^{-1} : Y \to X$ such that $f^{-1}(f(x)) = x$ for all $x \in X$.

This means $f$ must be one-to-one and onto: for every $y \in Y$ there is a unique $x \in X$ such that $f(x) = y$.

# Permutations versus functions example

Consider the following two functions $f\colon \{0,1\}^2 \to \{0,1\}^2$, where $X = Y = \{0,1\}^2$:

| $x$ | 00 | 01 | 10 | 11 |
|---|---|---|---|---|
| $f(x)$ | 01 | 11 | 00 | 10 |

A permutation

| $x$ | 00 | 01 | 10 | 11 |
|---|---|---|---|---|
| $f(x)$ | 01 | 11 | 11 | 10 |

Not a permutation

# Permutations versus functions example

Consider the following two functions $f\colon \{0,1\}^2 \to \{0,1\}^2$, where $X = Y = \{0,1\}^2$:

| $x$ | 00 | 01 | 10 | 11 |
|---|---|---|---|---|
| $f(x)$ | 01 | 11 | 00 | 10 |

A permutation

| $x$ | 00 | 01 | 10 | 11 |
|---|---|---|---|---|
| $f(x)$ | 01 | 11 | 11 | 10 |

Not a permutation

| $x$ | 00 | 01 | 10 | 11 |
|---|---|---|---|---|
| $f^{-1}(x)$ | 10 | 00 | 11 | 01 |

Its inverse

# Permutations versus functions example

Consider the following two functions $f\colon \{0,1\}^2 \to \{0,1\}^2$, where $X = Y = \{0,1\}^2$:

| $x$ | 00 | 01 | 10 | 11 |
|---|---|---|---|---|
| $f(x)$ | 01 | 11 | 00 | 10 |

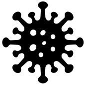
A permutation

| $x$ | 00 | 01 | 10 | 11 |
|---|---|---|---|---|
| $f^{-1}(x)$ | 10 | 00 | 11 | 01 |

Its inverse

| $x$ | 00 | 01 | 10 | 11 |
|---|---|---|---|---|
| $f(x)$ | 01 | 11 | 11 | 10 |

Not a permutation



No inverse, of course!

# Plan

Recall from last lecture

Notations, definitions

Definition of a block cipher

The DES block cipher

Two examples of formal attack scenarios

# Function families

## Definition: family of functions

A family of functions (also called a function family) is a two-input function

$$F : \text{Keys} \times \text{D} \to \text{R}.$$

Notation: For $K \in$ Keys we let

$$F_K : \begin{cases} \text{D} & \to & \text{R} \\ x & \mapsto & F(K, x) \end{cases}$$

In other words, $F_K(x) = F(K, x)$ for any $x \in$ D.

- The set Keys is called the key space.
  If Keys $= \{0,1\}^k$ we call $k$ the key length.
- The set D is called the input space.
  If D $= \{0,1\}^\ell$ we call $\ell$ the input length.
- The set R is called the output space, or range.
  If R $= \{0,1\}^L$ we call $L$ the output length.

# Example of a function family

**Example:** Define $F\colon \mathbb{Z}_3 \times \mathbb{Z}_3 \to \mathbb{Z}_3$ by $F(K, x) = (K \cdot x) \bmod 3$.

- This is a family of functions with domain $\mathbb{Z}_3 \times \mathbb{Z}_3$ and range $\mathbb{Z}_3$.
- If $K = 1$ then $F_1\colon \mathbb{Z}_3 \to \mathbb{Z}_3$ is given by $F_K(x) = x \bmod 3$.

# Block ciphers: Definition

## Definition: block cipher

Let $E$: Keys $\times$ D $\to$ R be a family of functions.
We say that $E$ is a block cipher if

- R $=$ D, meaning the input and output spaces are the same set.
- $E_K : \mathrm{D} \to \mathrm{D}$ is a permutation for every key $K \in$ Keys, meaning has an inverse $E_K^{-1} : \mathrm{D} \to \mathrm{D}$ such that $E_K^{-1}(E_K(x)) = x$ for all $x \in \mathrm{D}$.

We let $E^{-1}$: Keys $\times$ D $\to$ D, defined by $E^{-1}(K, y) = E_K^{-1}(y)$, be the inverse block cipher to $E$.

In practice we want that $E, E^{-1}$ are efficiently computable.

If Keys $= \{0,1\}^k$ then $k$ is the key length as before.
If R $=$ D $= \{0,1\}^\ell$ we call $\ell$ the block length.

# Block ciphers: Example

Block cipher $E$: $\{0,1\}^2 \times \{0,1\}^2 \to \{0,1\}^2$ (left), where the table entry corresponding to the key in row $K$ and input in column $x$ is $E_K(x)$. Its inverse $E^{-1}$: $\{0,1\}^2 \times \{0,1\}^2 \to \{0,1\}^2$ (right).

Keys:

| | 00 | 01 | 10 | 11 |
|---|---|---|---|---|
| 00 | 11 | 00 | 10 | 01 |
| 01 | 11 | 10 | 01 | 00 |
| 10 | 10 | 11 | 00 | 01 |
| 11 | 11 | 00 | 10 | 01 |

| | 00 | 01 | 10 | 11 |
|---|---|---|---|---|
| 00 | 01 | 11 | 10 | 00 |
| 01 | 11 | 10 | 01 | 00 |
| 10 | 10 | 11 | 00 | 01 |
| 11 | 01 | 11 | 10 | 00 |

- Row 01 of $E$ equals Row 01 of $E^{-1}$, meaning $E_{01} = E_{01}^{-1}$
- Rows have no repeated entries, for both $E$ and $E^{-1}$
- Column 00 of $E$ has repeated entries, that's ok
- Rows 00 and 11 of $E$ are the same, that's ok

# Block Ciphers: Example

Let $\ell = k$ and define $E\colon \{0,1\}^k \times \{0,1\}^\ell \to \{0,1\}^\ell$ by

$$E_K(x) = E(K,x) = K \oplus x$$

Then $E_K$ has inverse $E_K^{-1}$ where

$$E_K^{-1}(y) = K \oplus y$$

Why? Because

$$E_K^{-1}(E_K(x)) = E_K^{-1}(K \oplus x) = K \oplus K \oplus x = x$$

The inverse of block cipher $E$ is the block cipher $E^{-1}$ defined by

$$E^{-1}(K,y) = E_K^{-1}(y) = K \oplus y$$

# Exercise

Let $E$: Keys $\times$ D $\rightarrow$ D be a block cipher. Is $E$ a permutation?

- YES
- NO
- QUESTION DOESN'T MAKE SENSE
- WHO CARES?

This is an exercise in correct mathematical language.

# Exercise

Let $E$: Keys $\times$ D $\rightarrow$ D be a block cipher. Is $E$ a permutation?

**How to answer this:**

- Look back at the definition of a block cipher.
- Look back at the definition of a permutation.
- Pattern match these.
- Now come to a conclusion.

# Exercise

Above we had given the following example of a family of functions:

$$F : \begin{cases} \mathbb{Z}_3 \times \mathbb{Z}_3 & \rightarrow & \mathbb{Z}_3 \\ (K, x) & \mapsto & (K \cdot x) \bmod 3. \end{cases}$$

**Question:** Is $F$ a block cipher? Why or why not?

# Exercise

Above we had given the following example of a family of functions:

$$F : \begin{cases} \mathbb{Z}_3 \times \mathbb{Z}_3 & \rightarrow & \mathbb{Z}_3 \\ (K, x) & \mapsto & (K \cdot x) \bmod 3. \end{cases}$$

**Question:** Is $F$ a block cipher? Why or why not?

**Answer:** No, because $F_0(1) = F_0(2)$ so $F_0$ is not a permutation.

# Exercise

Above we had given the following example of a family of functions:

$$F : \begin{cases} \mathbb{Z}_3 \times \mathbb{Z}_3 & \rightarrow & \mathbb{Z}_3 \\ (K, x) & \mapsto & (K \cdot x) \bmod 3. \end{cases}$$

**Question:** Is $F$ a block cipher? Why or why not?

**Answer:** No, because $F_0(1) = F_0(2)$ so $F_0$ is not a permutation.

**Question:** Is $F_1$ a permutation?

# Exercise

Above we had given the following example of a family of functions:

$$F : \begin{cases} \mathbb{Z}_3 \times \mathbb{Z}_3 & \to & \mathbb{Z}_3 \\ (K, x) & \mapsto & (K \cdot x) \bmod 3. \end{cases}$$

**Question:** Is $F$ a block cipher? Why or why not?

**Answer:** No, because $F_0(1) = F_0(2)$ so $F_0$ is not a permutation.

**Question:** Is $F_1$ a permutation?

**Answer:** Yes. But that alone does not make $F$ a block cipher.

# A small modification

We now look at the very similar family of functions:

$$F : \begin{cases} \{1,2\} \times \mathbb{Z}_3 & \rightarrow & \mathbb{Z}_3 \\ (K, x) & \mapsto & (K \cdot x) \bmod 3. \end{cases}$$

The set of Keys is just $\mathrm{Keys} = \{1, 2\}$.

- The function $F_1$ is a

# A small modification

We now look at the very similar family of functions:

$$F : \begin{cases} \{1,2\} \times \mathbb{Z}_3 & \rightarrow & \mathbb{Z}_3 \\ (K,x) & \mapsto & (K \cdot x) \bmod 3. \end{cases}$$

The set of Keys is just $\text{Keys} = \{1,2\}$.

- The function $F_1$ is a permutation.
- The function $F_2$ is a

# A small modification

We now look at the very similar family of functions:

$$F : \begin{cases} \{1,2\} \times \mathbb{Z}_3 & \rightarrow & \mathbb{Z}_3 \\ (K, x) & \mapsto & (K \cdot x) \bmod 3. \end{cases}$$

The set of Keys is just Keys $= \{1, 2\}$.

- The function $F_1$ is a permutation.
- The function $F_2$ is a permutation.

Therefore $F$ defines a block cipher.

(a very simplistic one!)

# Plan

Recall from last lecture

Notations, definitions

Definition of a block cipher

The DES block cipher

Two examples of formal attack scenarios

# Block cipher usage

Let $E\colon \{0,1\}^k \times \{0,1\}^\ell \to \{0,1\}^\ell$ be a block cipher.

The block cipher $E$ is considered public (Kerckhoffs). In typical usage:

- $K \xleftarrow{\$} \{0,1\}^k$ is known to parties $S$ (sender) and $R$ (receiver), but the key $K$ is not given to adversary $A$.
- $S$ uses $E_K$ for encryption , $R$ uses $E_K^{-1}$ for decryption

$S$

$y_1, y_2, \ldots$

Adversary $A$

$R$

$y_i = E_K(x_i)$
$i = 1, 2, \ldots$

$x_i = E_K^{-1}(y_i)$
$i = 1, 2, \ldots$

Leads to security requirements like:

- Hard to get $K$ from $y_1, y_2, \ldots$;
- Hard to get $x_i$ from $y_i$; ...

# DES History

1972 – NBS (now NIST) asked for a block cipher for standardization

1974 – IBM designs Lucifer

Lucifer eventually evolved into DES.

Widely adopted as a standard including by ANSI and American Bankers association

Used in ATM machines

Replaced (by AES) in 2001.

# DES parameters

Key Length $k = 56$

Block length $\ell = 64$

So,

$$\mathrm{DES}\colon \{0,1\}^{56} \times \{0,1\}^{64} \to \{0,1\}^{64}$$

$$\mathrm{DES}^{-1}\colon \{0,1\}^{56} \times \{0,1\}^{64} \to \{0,1\}^{64}$$

DES is a block cipher: for any $k \in \mathrm{Keys} = \{0,1\}^{56}$, the function $\mathrm{DES}_k$ is a permutation.

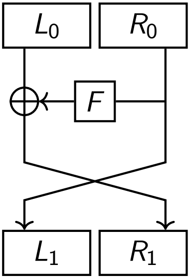
# DES construction

Several important concepts are present in the construction of DES:

- DES is a Feistel network, made of several successive rounds.
- Each round performs a simple operation.
- Something that is derived from the key is used at each round, via a Key schedule algorithm.
- Most of the structure resembles a linear function, but nonlinearity is inserted at very important places.
- Non-linearity is done by small table lookups called $S$-boxes.

Nowadays, DES is obsolete, but its design concepts are still relevant today.

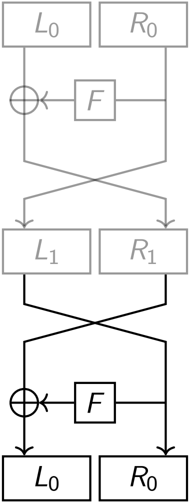
# One round in a Feistel network (in DES)




- $L_0, R_0$ are bitstrings of equal length: 32 bits.
- $F$ is some nonlinear function. $F$ does not have to be a permutation.
- We have constructed a function

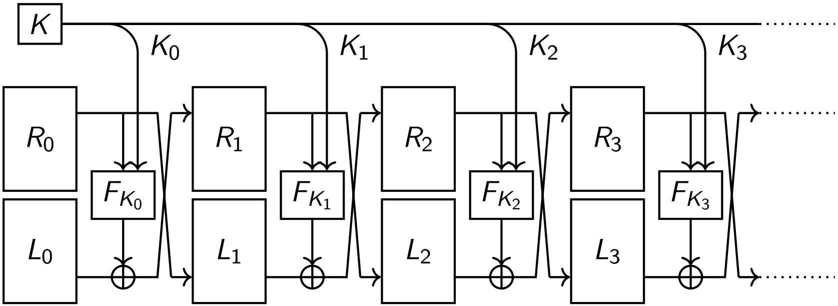
$$\mathcal{R}_F : \begin{cases} \{0,1\}^{64} & \to & \{0,1\}^{64} \\ (L_0, R_0) & \mapsto & (R_0, L_0 \oplus F(R_0)) \end{cases}$$

# We can invert one round quite easily




Because of this simple fact, one round $\mathcal{R}_F$ is a permutation, whatever the function $F$. We use it to create a block cipher.

# Chaining multiple rounds




- One round is a pretty simple permutation, but chaining them one after another makes the resulting permutation a lot more complicated.
- In DES, as many as 16 rounds are chained to form a block cipher.

# DES Construction

function $\mathsf{DES}_K(M)$ // $|K| = 56$ and $|M| = 64$

$(K_1, \ldots, K_{16}) \leftarrow KeySchedule(K)$ // $|K_i| = 48$ for $1 \le i \le 16$

$M \leftarrow IP(M)$ // initial permutation

Parse $M$ as $L_0 \parallel R_0$ // $|L_0| = |R_0| = 32$

for $i = 1$ to $16$ do

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

$C \leftarrow IP^{-1}(L_{16} \parallel R_{16})$

return $C$

function $\mathsf{DES}_K^{-1}(C)$ // $|K| = 56$ and $|M| = 64$

$(K_1, \ldots, K_{16}) \leftarrow KeySchedule(K)$ // $|K_i| = 48$ for $1 \le i \le 16$

$C \leftarrow IP(C)$

Parse $C$ as $L_{16} \parallel R_{16}$

for $i = 16$ downto $1$ do

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

$M \leftarrow IP^{-1}(L_0 \parallel R_0)$

return $M$

# DES Construction

function $\mathsf{DES}_K(M)$ // $|K| = 56$ and $|M| = 64$

$(K_1, \ldots, K_{16}) \leftarrow \mathit{KeySchedule}(K)$ // $|K_i| = 48$ for $1 \leq i \leq 16$

$M \leftarrow IP(M)$

Parse $M$ as $L_0 \,\|\, R_0$ // $|L_0| = |R_0| = 32$

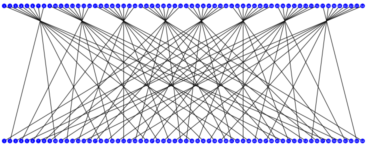
for $i = 1$ to $16$ do

$L_i \leftarrow R_{i-1}$ ; $R_i \leftarrow f(K_i, R_{i-1}) \oplus L_{i-1}$

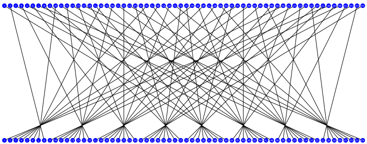
$C \leftarrow IP^{-1}(L_{16} \,\|\, R_{16})$

return $C$

Initial permutation: given explicitly by a table (see Wikipedia).



$IP$



$IP^{-1}$

# DES Construction

function $F(J, R)$ // $|J| = 48$ and $|R| = 32$

$R \leftarrow E(R)$ ; $R \leftarrow R \oplus J$

Parse $R$ as $R_1 \| R_2 \| R_3 \| R_4 \| R_5 \| R_6 \| R_7 \| R_8$ // $|R_i| = 6$ for $1 \leq i \leq$

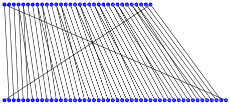
for $i = 1, \ldots, 8$ do

$R_i \leftarrow \mathbf{S}_i(R_i)$ // Each S-box returns 4 bits

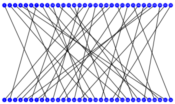
$R \leftarrow R_1 \| R_2 \| R_3 \| R_4 \| R_5 \| R_6 \| R_7 \| R_8$ // $|R| = 32$ bits

$R \leftarrow P(R)$ ; return $R$

Expansion $E$ and permutation $P$ are given explicitly by tables (see Wikipedia).



$E$



$P$

# S-boxes

All S-boxes are nonlinear function with 6-bit inputs and 4-bit outputs. They are given explicitly by tables (again, see Wikipedia).

- The minimal size of these tables is totally understandable given the implementation constraints of the time. 8 tables with 64 values of 4 bits each means a quarter of a kilobyte, and that was something, in the 1970s!
- How the values in the tables were chosen remained a mystery for many years.

# Plan

Recall from last lecture

Notations, definitions

Definition of a block cipher

The DES block cipher

Two examples of formal attack scenarios

# Key Recovery Attack Scenario

Let $E$: Keys $\times$ D $\rightarrow$ R be a block cipher known to the adversary $A$.

- Sender Alice and receiver Bob share a *target key* $K \in$ Keys.
- Alice encrypts $M_i$ to get $C_i = E_K(M_i)$ for $1 \leq i \leq q$, and transmits $C_1, \ldots, C_q$ to Bob
- The adversary gets $C_1, \ldots, C_q$ and also knows $M_1, \ldots, M_q$
- Now the adversary wants to figure out $K$ so that it can decrypt any future ciphertext $C$ to recover $M = E_K^{-1}(C)$.

**Question:** Why do we assume $A$ knows $M_1, \ldots, M_q$?

**Answer:** Reasons include a posteriori revelation of data, a priori knowledge of context, and just being conservative!

# Key Recovery Security Metrics

We consider two measures (metrics) for how well the adversary does at this key recovery task:

- Target key recovery (TKR)
- Consistent key recovery (KR)

In each case the definition involves a game and an advantage.

The definitions will allow $E$ to be any family of functions, not just a block cipher.

The definitions allow $A$ to pick, not just know, $M_1, \ldots, M_q$. This is called a chosen-plaintext attack.

# Target Key Recovery: The game

**Game $\mathrm{TKR}_E$**

**procedure Initialize**
$K \xleftarrow{\$} \text{Keys}$

**procedure Fn**$(M)$
Return $E(K, M)$

**procedure Finalize**$(K')$
Return $(K = K')$

- First **Initialize** executes, selecting *target key* $K \xleftarrow{\$} \text{Keys}$, but not giving it to $A$.
- Now $A$ can call (query) **Fn** on any input $M \in \mathrm{D}$ of its choice to get back $C = E_K(M)$. It can make as many queries as it wants.

$$\text{queries } M_1, \ldots, M_q \rightarrow \text{answers } C_1, \ldots, C_q.$$

- Eventually $A$ will halt with an output $K'$ which is automatically viewed as the input to **Finalize**
- The game returns whatever **Finalize** returns

# Common notations

## Notations: games

- $\mathrm{TKR}$ is a game. It includes some randomness.
- It is parameterized by something. Here, it is a block cipher. We speak of the game $\mathrm{TKR}_E$ the parameter
- Some player (program) $A$ will play the game. The game can return True or False. Whether $A$ succeeds or not is $\mathrm{TKR}_E^A$. the adversary

## Notation: advantages

We define some advantages, that are related to some games:

- **Adv** is our generic notation for an advantage.
  $\mathbf{Adv}^{\mathrm{tkr}}$, for example is related to the game $\mathrm{TKR}$.
- $\mathbf{Adv}_E^{\mathrm{tkr}}$ is related to the game $\mathrm{TKR}_E$, parameterized by $E$.
- $\mathbf{Adv}_E^{\mathrm{tkr}}(A)$ is related to how well $A$ performs when playing $\mathrm{TKR}_E$.

# Definition of $\textbf{Adv}^{\mathrm{tkr}}$

**Game $\mathrm{TKR}_E$**

**procedure Initialize**
$K \xleftarrow{\$} \mathrm{Keys}$

**procedure Fn**$(M)$
Return $E(K, M)$

**procedure Finalize**$(K')$
Return $(K = K')$

## Definition of $\textbf{Adv}^{\mathrm{tkr}}$

$\textbf{Adv}^{\mathrm{tkr}}$ is defined from the game $\mathrm{TKR}$:

$$\textbf{Adv}^{\mathrm{tkr}}_E(A) = \Pr[\mathrm{TKR}_E^A \Rightarrow \mathrm{true}].$$

The tkr advantage of $A$ is the probability that the game $\mathrm{TKR}$ returns true

# Consistent keys

**Definition: consistent keys**

Let $E$: Keys $\times$ D $\rightarrow$ R be a family of functions. We say that key $K' \in$ Keys is *consistent* with $(M_1, C_1), \ldots, (M_q, C_q)$ if $E(K', M_i) = C_i$ for all $1 \leq i \leq q$.

**Example:** For $E$: $\{0,1\}^2 \times \{0,1\}^2 \rightarrow \{0,1\}^2$ defined by

| | 00 | 01 | 10 | 11 |
|---|---|---|---|---|
| 00 | 11 | 00 | 10 | 01 |
| 01 | 11 | 10 | 01 | 00 |
| 10 | 10 | 11 | 00 | 01 |
| 11 | 11 | 00 | 10 | 01 |

The entry in row $K$, column $M$ is $E(K, M)$.

- Key 00 is consistent with $(11, 01)$
- Key 10 is consistent with $(11, 01)$
- Key 00 is consistent with $(01, 00), (11, 01)$
- Key 11 is consistent with $(01, 00), (11, 01)$

# Consistent Key Recovery: Game and Advantage

Let $E$: Keys $\times$ D $\rightarrow$ R be a family of functions, and $A$ an adversary.

**Game** $\mathrm{KR}_E$

**procedure Initialize**
$K \xleftarrow{\$}$ Keys

**procedure Fn**$(M)$
Return $E(K, M)$

**procedure Finalize**$(K')$
For $j = 1, \ldots, i$ do
If $E(K', M_j) \neq C_j$ then Return false
If $M_j \in \{M_1, \ldots, M_{j-1}\}$ then Return false
Return true

The game returns true if (1) The key $K'$ returned by the adversary is consistent with $(M_1, C_1), \ldots, (M_q, C_q)$, and (2) $M_1, \ldots, M_q$ are distinct.

$A$ is a $q$-query adversary if it makes $q$ distinct queries to its **Fn** oracle.

## Definition of $\mathbf{Adv}^{\mathrm{kr}}$

$$\mathbf{Adv}^{\mathrm{kr}}_E(A) = \Pr[\mathrm{KR}^A_E \Rightarrow \text{true}].$$

# kr advantage always exceeds tkr advantage

**Fact:** Suppose that, in game $\mathrm{KR}_E$, adversary $A$ makes queries $M_1, \ldots,$ $M_q$ to **Fn**, thereby defining $C_1, \ldots, C_q$. Then the target key $K$ is consistent with $(M_1, C_1), \ldots, (M_q, C_q)$.

**Proposition:** Let $E$ be a family of functions. Let $A$ be *any* adversary all of whose **Fn** queries are distinct. Then

$$\mathbf{Adv}_E^{\mathrm{kr}}(A) \geq \mathbf{Adv}_E^{\mathrm{tkr}}(A) .$$

**Why?** If the $K'$ that $A$ returns equals the target key $K$, then, by the Fact, the input-output examples $(M_1, C_1), \ldots, (M_q, C_q)$ will of course be consistent with $K'$.

# Impact of the number of queries

Another comparison: same game, but adversaries that differ in the number of queries they make.

Doing more queries in the tkr (target key recovery) game makes it:

☐ Easier. ☐ Harder. ☐ It depends.

# Impact of the number of queries

Another comparison: same game, but adversaries that differ in the number of queries they make.

Doing more queries in the tkr (target key recovery) game makes it:

■ Easier. ~~☐ Harder.~~ ~~☐ It depends.~~

(but the difference can be very close to zero!)

Doing more queries in the kr (consistent key recovery) game makes it:

☐ Easier. ☐ Harder. ☐ It depends.

# Impact of the number of queries

Another comparison: same game, but adversaries that differ in the number of queries they make.

Doing more queries in the tkr (target key recovery) game makes it:

■ Easier. ~~□ Harder.~~ ~~□ It depends.~~

(but the difference can be very close to zero!)

Doing more queries in the kr (consistent key recovery) game makes it:

~~□ Easier.~~ ~~□ Harder.~~ ■ It depends.

(harder for trivial adversaries. Becomes easier later when $\mathbf{Adv}^{\mathrm{tkr}}$ starts to increase)