BLOCK CIPHERS
A function $f : \{0, 1\}^\ell \rightarrow \{0, 1\}^\ell$ is a **permutation** if there is an inverse function $f^{-1} : \{0, 1\}^\ell \rightarrow \{0, 1\}^\ell$ satisfying

$$\forall x \in \{0, 1\}^\ell : f^{-1}(f(x)) = x$$

This means $f$ must be one-to-one and onto, meaning for every $y \in \{0, 1\}^\ell$ there is a unique $x \in \{0, 1\}^\ell$ such that $f(x) = y$. 
Permutations and Inverses

<table>
<thead>
<tr>
<th>$x$</th>
<th>00</th>
<th>01</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f(x)$</td>
<td>01</td>
<td>11</td>
<td>00</td>
<td>10</td>
</tr>
</tbody>
</table>

A permutation

<table>
<thead>
<tr>
<th>$x$</th>
<th>00</th>
<th>01</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f(x)$</td>
<td>01</td>
<td>11</td>
<td>11</td>
<td>10</td>
</tr>
</tbody>
</table>

Not a permutation
Permutations and Inverses

<table>
<thead>
<tr>
<th>$x$</th>
<th>00</th>
<th>01</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f(x)$</td>
<td>01</td>
<td>11</td>
<td>00</td>
<td>10</td>
</tr>
</tbody>
</table>

A permutation

<table>
<thead>
<tr>
<th>$x$</th>
<th>00</th>
<th>01</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f^{-1}(x)$</td>
<td>10</td>
<td>00</td>
<td>11</td>
<td>01</td>
</tr>
</tbody>
</table>

Its inverse
Let
\[ E : \{0, 1\}^k \times \{0, 1\}^\ell \rightarrow \{0, 1\}^\ell \]
be a function taking a key \( K \) and input \( x \) to return output \( E(K, x) \). For each key \( K \) we let \( E_K : \{0, 1\}^\ell \rightarrow \{0, 1\}^\ell \) be the function defined by
\[ E_K(x) = E(K, x) . \]
We say that \( E \) is a block cipher if

- \( E_K : \{0, 1\}^\ell \rightarrow \{0, 1\}^\ell \) is a permutation for every \( K \), meaning has an inverse \( E_K^{-1} \),
- \( E, E^{-1} \) are efficiently computable,

where \( E^{-1}(K, x) = E_K^{-1}(x) \).
Example

The table entry corresponding to the key in row $K$ and input in column $x$ is $E_K(x)$.

<table>
<thead>
<tr>
<th></th>
<th>00</th>
<th>01</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>00</td>
<td>01</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>01</td>
<td>01</td>
<td>00</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>11</td>
<td>00</td>
<td>01</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>10</td>
<td>01</td>
<td>00</td>
</tr>
</tbody>
</table>

In this case, the inverse cipher $E^{-1}$ is given by the same table: the table entry corresponding to the key in row $K$ and output in column $y$ is $E_K^{-1}(y)$. 
Let $\ell = k$ and define $E : \{0, 1\}^k \times \{0, 1\}^\ell \to \{0, 1\}^\ell$ 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

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

- \( K \leftarrow \{0, 1\}^k \)
- \( K \) (magically) given to parties S, R, but not to A.
- S,R use \( E_K \)

Algorithm \( E \) is public! Think of \( E_K \) as encryption under key \( K \).

![Diagram showing block cipher usage](image)

Leads to security requirements like:

- Hard to get \( K \) from \( y_1, y_2, \ldots \)
- Hard to get \( x_i \) from \( y_i \)
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.
Key Length $k = 56$

Block length $\ell = 64$

So,

$$\text{DES: } \{0, 1\}^{56} \times \{0, 1\}^{64} \rightarrow \{0, 1\}^{64}$$

$$\text{DES}^{-1}: \{0, 1\}^{56} \times \{0, 1\}^{64} \rightarrow \{0, 1\}^{64}$$
DES Construction

function $\text{DES}_K(M)$ \hspace{1mm} // $|K| = 56$ and $|M| = 64$

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

$M \leftarrow \text{IP}(M)$

Parse $M$ as $L_0 \parallel R_0$ \hspace{1mm} // $|L_0| = |R_0| = 32$

for $i = 1$ to $16$ do

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

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

return $C$

Round $i$: Invertible given $K_i$: 
function DES\(_K(M)\) \hspace{1em} // \ |K| = 56 and \ |M| = 64
\((K_1, \ldots, K_{16}) \leftarrow \text{KeySchedule}(K)\) \hspace{1em} // \ |K_i| = 48 for 1 \leq i \leq 16
\(M \leftarrow IP(M)\)
Parse \(M\) as \(L_0 \parallel R_0\) \hspace{1em} // \ |L_0| = |R_0| = 32
for \(i = 1\) to 16 do
    \(L_i \leftarrow R_{i-1}\); \hspace{1em} \(R_i \leftarrow f(K_i, R_{i-1}) \oplus L_{i-1}\)
\(C \leftarrow IP^{-1}(L_{16} \parallel R_{16})\)
return \(C\)

function DES\(^{-1}\_K(C)\) \hspace{1em} // \ |K| = 56 and \ |M| = 64
\((K_1, \ldots, K_{16}) \leftarrow \text{KeySchedule}(K)\) \hspace{1em} // \ |K_i| = 48 for 1 \leq i \leq 16
\(C \leftarrow IP(C)\)
Parse \(C\) as \(L_{16} \parallel R_{16}\)
for \(i = 16\) downto 1 do
    \(R_{i-1} \leftarrow L_i\); \hspace{1em} \(L_{i-1} \leftarrow f(K_i, R_{i-1}) \oplus R_i\)
\(M \leftarrow IP^{-1}(L_0 \parallel R_0)\)
return \(M\)
function $\text{DES}_K(M)$  // $|K| = 56$ and $|M| = 64$

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

$M \leftarrow \text{IP}(M)$

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

for $i = 1$ to $16$ do

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

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

return $C$

\[
\begin{array}{cccccccccccccccc}
58 & 50 & 42 & 34 & 26 & 18 & 10 & 2 & & & & & & & & \\
60 & 52 & 44 & 36 & 28 & 20 & 12 & 4 & & & & & & & & \\
62 & 54 & 46 & 38 & 30 & 22 & 14 & 6 & & & & & & & & \\
64 & 56 & 48 & 40 & 32 & 24 & 16 & 8 & & & & & & & & \\
57 & 49 & 41 & 33 & 25 & 17 & 9 & 1 & & & & & & & & \\
59 & 51 & 43 & 35 & 27 & 19 & 11 & 3 & & & & & & & & \\
61 & 53 & 45 & 37 & 29 & 21 & 13 & 5 & & & & & & & & \\
\end{array}
\]

\[
\begin{array}{cccccccccccccccc}
40 & 8 & 48 & 16 & 56 & 24 & 64 & 32 & & & & & & & & \\
38 & 6 & 46 & 14 & 54 & 22 & 62 & 30 & & & & & & & & \\
37 & 5 & 45 & 13 & 53 & 21 & 61 & 29 & & & & & & & & \\
36 & 4 & 44 & 12 & 52 & 20 & 60 & 28 & & & & & & & & \\
35 & 3 & 43 & 11 & 51 & 19 & 59 & 27 & & & & & & & & \\
34 & 2 & 42 & 10 & 50 & 18 & 58 & 26 & & & & & & & & \\
33 & 1 & 41 & 9 & 49 & 17 & 57 & 25 & & & & & & & & \\
\end{array}
\]
function $f(J, R)$  // $|J| = 48$ and $|R| = 32$

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

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

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

$R_i \leftarrow S_i(R_i)$  // Each S-box returns 4 bits

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

$R \leftarrow P(R)$

return $R$

\[
\begin{array}{cccccc}
E & 32 & 1 & 2 & 3 & 4 & 5 \\
4 & 5 & 6 & 7 & 8 & 9 \\
8 & 9 & 10 & 11 & 12 & 13 \\
12 & 13 & 14 & 15 & 16 & 17 \\
16 & 17 & 18 & 19 & 20 & 21 \\
20 & 21 & 22 & 23 & 24 & 25 \\
24 & 25 & 26 & 27 & 28 & 29 \\
28 & 29 & 30 & 31 & 32 & 1 \\
\end{array}
\]

\[
\begin{array}{cccccc}
P & 16 & 7 & 20 & 21 \\
29 & 12 & 28 & 17 \\
1 & 15 & 23 & 26 \\
5 & 18 & 31 & 10 \\
2 & 8 & 24 & 14 \\
32 & 27 & 3 & 9 \\
19 & 13 & 30 & 6 \\
22 & 11 & 4 & 25 \\
\end{array}
\]
S-boxes

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>0 0</td>
<td>14</td>
<td>4</td>
<td>13</td>
<td>1</td>
<td>2</td>
<td>15</td>
<td>11</td>
<td>8</td>
<td>3</td>
<td>10</td>
<td>6</td>
<td>12</td>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>0 1</td>
<td>0</td>
<td>15</td>
<td>7</td>
<td>4</td>
<td>14</td>
<td>2</td>
<td>13</td>
<td>1</td>
<td>10</td>
<td>6</td>
<td>12</td>
<td>11</td>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>1 0</td>
<td>4</td>
<td>1</td>
<td>14</td>
<td>8</td>
<td>13</td>
<td>6</td>
<td>2</td>
<td>11</td>
<td>15</td>
<td>12</td>
<td>9</td>
<td>7</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>1 1</td>
<td>15</td>
<td>12</td>
<td>8</td>
<td>2</td>
<td>4</td>
<td>9</td>
<td>1</td>
<td>7</td>
<td>5</td>
<td>11</td>
<td>3</td>
<td>14</td>
<td>10</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_2$</td>
<td>0 0</td>
<td>15</td>
<td>1</td>
<td>8</td>
<td>14</td>
<td>6</td>
<td>11</td>
<td>3</td>
<td>4</td>
<td>9</td>
<td>7</td>
<td>2</td>
<td>13</td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0 1</td>
<td>3</td>
<td>13</td>
<td>4</td>
<td>7</td>
<td>15</td>
<td>2</td>
<td>8</td>
<td>14</td>
<td>12</td>
<td>0</td>
<td>1</td>
<td>10</td>
<td>6</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>1 0</td>
<td>0</td>
<td>14</td>
<td>7</td>
<td>11</td>
<td>10</td>
<td>4</td>
<td>13</td>
<td>1</td>
<td>5</td>
<td>8</td>
<td>12</td>
<td>6</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>1 1</td>
<td>13</td>
<td>8</td>
<td>10</td>
<td>1</td>
<td>3</td>
<td>15</td>
<td>4</td>
<td>2</td>
<td>11</td>
<td>6</td>
<td>7</td>
<td>12</td>
<td>0</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_3$</td>
<td>0 0</td>
<td>10</td>
<td>0</td>
<td>9</td>
<td>14</td>
<td>6</td>
<td>3</td>
<td>15</td>
<td>5</td>
<td>1</td>
<td>13</td>
<td>12</td>
<td>7</td>
<td>11</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>0 1</td>
<td>13</td>
<td>7</td>
<td>0</td>
<td>9</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td>10</td>
<td>2</td>
<td>8</td>
<td>5</td>
<td>14</td>
<td>12</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>1 0</td>
<td>13</td>
<td>6</td>
<td>4</td>
<td>9</td>
<td>8</td>
<td>15</td>
<td>3</td>
<td>0</td>
<td>11</td>
<td>1</td>
<td>2</td>
<td>12</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>1 1</td>
<td>1</td>
<td>10</td>
<td>13</td>
<td>0</td>
<td>6</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>4</td>
<td>15</td>
<td>14</td>
<td>3</td>
<td>11</td>
<td>5</td>
</tr>
</tbody>
</table>

Figure: The DES S-boxes.
Let $E: \{0, 1\}^k \times \{0, 1\}^\ell \to \{0, 1\}^\ell$ be a blockcipher. It is known to the adversary $A$.

**Def:** We say that $K' \in \{0, 1\}^k$ is *consistent* with $(M_1, C_1), \ldots, (M_q, C_q)$ if $E(K', M_i) = C_i$ for all $1 \leq i \leq q$.

**Key-recovery security game, informally:**

- A target key $K \leftarrow^\$ \{0, 1\}^k$ is selected but not given to $A$.
- $A$ can submit a plaintext $M \in \{0, 1\}^\ell$ and get back $C = E(K, M)$, in this way gathering input-output examples $(M_1, C_1), \ldots, (M_q, C_q)$ of $E(K, \cdot)$.
- $A$ outputs a “guess” $K'$
- $A$ wins if $K'$ is consistent with $(M_1, C_1), \ldots, (M_q, C_q)$. 
Key-recovery security game, informally:

- A target key \( K \leftarrow \{0,1\}^k \) is selected but not given to \( A \).
- \( A \) can submit a plaintext \( M \in \{0,1\}^\ell \) and get back \( C = E(K,M) \), in this way gathering input-output examples \((M_1,C_1),\ldots,(M_q,C_q)\) of \( E(K,\cdot) \).
- \( A \) outputs a “guess” \( K' \)
- \( A \) wins if \( K' \) is consistent with \((M_1,C_1),\ldots,(M_q,C_q)\).

Usually, if \( K' \) is consistent with \( K \), then \( K' = K \), so the attack recovers the target key.

**About the model:** Certainly \( A \) should be given \( C_1,\ldots,C_q \). But why does \( A \) get to pick \( M_1,\ldots,M_q \)? Reasons include a posteriori revelation of data, a priori knowledge of context, and just being conservative!
Let $E: \{0, 1\}^k \times \{0, 1\}^\ell \to \{0, 1\}^\ell$ be a blockcipher and $A$ an adversary.

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

**procedure Initialize**

$K \leftarrow \{0, 1\}^k; \ i \leftarrow 0$

**procedure $\text{Fn}(M)$**

$i \leftarrow i + 1; \ M_i \leftarrow M$

$C_i \leftarrow E(K, M_i); \ \text{Return } C_i$

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

$\text{win} \leftarrow \text{true}$

For $j = 1, \ldots, i$ do

If $E(K', M_i) \neq C_i$ then $\text{win} \leftarrow \text{false}$

Return $\text{win}$

$\text{Adv}_{\text{kr}}^E(A) = \Pr[\text{KR}_E^A \Rightarrow \text{true}]$
Running a game with an adversary

• First **Initialize** executes
• Now $A$ can call (query) $F_n$ on any input $M$ of its choice. It can make as many queries as it wants
• Eventually $A$ will halt with an output $K'$ which is automatically viewed as the input to **Finalize**
• The game returns whatever **Finalize** returns
• The advantage of $A$ is the probability that the game returns true

$\text{Adv}_{E}^{kr}(A)$ will depend on the number $q$ of queries that $A$ makes and its running time.
Exhaustive Key Search

Let $T_1, \ldots, T_{2^k}$ be a list of all $k$ bit keys and let $\langle i \rangle$ denote the $\ell$-bit binary representation of integer $i$. Let $1 \leq q \leq 2^\ell$ be a parameter.

**adversary $A_{eks}$**

For $j = 1, \ldots, q$ do $M_j \leftarrow \langle j - 1 \rangle$; $C_j \leftarrow \text{Fn}(M_j)$

For $i = 1, \ldots, 2^k$ do

if $(\forall j \in \{1, \ldots, q\} : E(T_i, M_j) = C_j)$ then return $T_i$

Then $\text{Adv}_E^{kr}(A_{eks}) = 1$ because $K \in \{T_1, \ldots, T_{2^k}\}$ and $K$ is consistent with $(M_1, C_1), \ldots, (M_q, C_q)$.

Think of $q$ as small, like $q \in \{1, 2, 3\}$. As long as $q > k/n$, empirical evidence says that the attack returns the target key $K$ itself.
How long does exhaustive key search take?

DES can be computed at 1.6 Gbits/sec in hardware.

DES plaintext = 64 bits

Chip can perform \((1.6 \times 10^9)/64 = 2.5 \times 10^7\) DES computations per second

Expect \(A_{eks} (q = 1)\) to succeed in \(2^{55}\) DES computations, so it takes time

\[
\frac{2^{55}}{2.5 \times 10^7} \approx 1.4 \times 10^9 \text{ seconds}
\]

\[
\approx 45 \text{ years!}
\]

Key Complementation \(\Rightarrow 22.5\) years

But this is prohibitive. Does this mean DES is secure?
Exhaustive key search is a generic attack: Did not attempt to “look inside” DES and find/exploit weaknesses.

The following non-generic key-recovery attacks on DES have advantage close to one and running time smaller than $2^{56}$ DES computations:

<table>
<thead>
<tr>
<th>Attack</th>
<th>when</th>
<th>$q$, running time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Differential cryptanalysis</td>
<td>1992</td>
<td>$2^{47}$</td>
</tr>
<tr>
<td>Linear cryptanalysis</td>
<td>1993</td>
<td>$2^{44}$</td>
</tr>
</tbody>
</table>
Exhaustive key search is a generic attack: Did not attempt to “look inside” DES and find/exploit weaknesses.  

The following non-generic key-recovery attacks on DES have advantage close to one and running time smaller than $2^{56}$ DES computations:

<table>
<thead>
<tr>
<th>Attack</th>
<th>when</th>
<th>$q$, running time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Differential cryptanalysis</td>
<td>1992</td>
<td>$2^{47}$</td>
</tr>
<tr>
<td>Linear cryptanalysis</td>
<td>1993</td>
<td>$2^{44}$</td>
</tr>
</tbody>
</table>

But merely storing $2^{44}$ input-output pairs requires 281 Tera-bytes.  
In practice these attacks were prohibitively expensive.
adversary $A_{ek}$

For $j = 1, \ldots, q$ do $M_j \leftarrow \langle j - 1 \rangle$; $C_j \leftarrow F_n(M_j)$

For $i = 1, \ldots, 2^k$ do

if $(\forall j \in \{1, \ldots, q\} : E(T_i, M_j) = C_j)$ then return $T_i$
adversary $A_{\text{eks}}$

For $j = 1, \ldots, q$ do $M_j \leftarrow \langle j - 1 \rangle$; $C_j \leftarrow \text{Fn}(M_j)$

For $i = 1, \ldots, 2^k$ do

if $(\forall j \in \{1, \ldots, q\} : E(T_i, M_j) = C_j)$ then return $T_i$

**Observation:** The $E$ computations can be performed in parallel!
adversary $A_{eks}$
For $j = 1, \ldots, q$ do $M_j \leftarrow \langle j - 1 \rangle$; $C_j \leftarrow F_n(M_j)$
For $i = 1, \ldots, 2^k$ do
  if $(\forall j \in \{1, \ldots, q\} : E(T_i, M_j) = C_j)$ then return $T_i$

Observation: The $E$ computations can be performed in parallel!

In 1993, Wiener designed a dedicated DES-cracking machine:
- $\$1 million
- 57 chips, each with many, many DES processors
- Finds key in 3.5 hours
DES security summary

DES is considered broken because its short key size permits rapid key-search.

But DES is a very strong design as evidenced by the fact that there are no practical attacks that exploit its structure.
Block cipher $2DES : \{0, 1\}^{112} \times \{0, 1\}^{64} \rightarrow \{0, 1\}^{64}$ is defined by

$$2DES_{K_1K_2}(M) = DES_{K_2}(DES_{K_1}(M))$$

- Exhaustive key search takes $2^{112}$ DES computations, which is too much even for machines
- Resistant to differential and linear cryptanalysis.
Suppose $K_1 K_2$ is a target 2DES key and adversary has $M, C$ such that

$$C = 2\text{DES}_{K_1 K_2}(M) = \text{DES}_{K_2}(\text{DES}_{K_1}(M))$$

Then

$$\text{DES}_{K_2}^{-1}(C) = \text{DES}_{K_1}(M)$$
Meet-in-the-middle attack on 2DES

Suppose $\text{DES}_{K_2}^{-1}(C) = \text{DES}_{K_1}(M)$ and $T_1, \ldots, T_N$ are all possible DES keys, where $N = 2^{56}$.

<table>
<thead>
<tr>
<th>$T_1$</th>
<th>$\text{DES}(T_1, M)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_i$</td>
<td>$\text{DES}(T_i, M)$</td>
</tr>
<tr>
<td>$T_N$</td>
<td>$\text{DES}(T_N, M)$</td>
</tr>
</tbody>
</table>

Table $L$

<table>
<thead>
<tr>
<th>$\text{DES}^{-1}(T_1, C)$</th>
<th>$T_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{DES}^{-1}(T_j, C)$</td>
<td>$T_j$</td>
</tr>
<tr>
<td>$\text{DES}^{-1}(T_N, C)$</td>
<td>$T_N$</td>
</tr>
</tbody>
</table>

Table $R$

Attack idea:

- Build L,R tables
Meet-in-the-middle attack on 2DES

Suppose $\text{DES}^{-1}_{K_2}(C) = \text{DES}_{K_1}(M)$ and $T_1, \ldots, T_N$ are all possible DES keys, where $N = 2^{56}$.

![Table L](image)

**Table L**

- $K_1 \rightarrow$
  - $T_1 \rightarrow \text{DES}(T_1, M)$
  - $T_i \rightarrow \text{DES}(T_i, M)$
  - $T_N \rightarrow \text{DES}(T_N, M)$

![Table R](image)

**Table R**

- $\text{DES}^{-1}(T_1, C) \rightarrow T_1$
  - $\text{DES}^{-1}(T_j, C) \rightarrow T_j$
  - $\text{DES}^{-1}(T_N, C) \rightarrow T_N$

**Attack idea:**

- Build L, R tables
- Find $i, j$ s.t. $L[i] = R[j]$
- Guess that $K_1K_2 = T_iT_j$
Meet-in-the-middle attack on 2DES

Let $T_1, \ldots, T_{2^{56}}$ denote an enumeration of DES keys.

**adversary** $A_{\text{MinM}}$

$M_1 \leftarrow 0^{64}; C_1 \leftarrow \text{Fn}(M_1)$

for $i = 1, \ldots, 2^{56}$ do $L[i] \leftarrow \text{DES}(T_i, M_1)$

for $j = 1, \ldots, 2^{56}$ do $R[j] \leftarrow \text{DES}^{-1}(T_j, C_1)$

$S \leftarrow \{(i,j) : L[i] = R[j]\}$

Pick some $(l, r) \in S$ and return $T_l \parallel T_r$

Attack takes about $2^{57}$ DES/DES$^{-1}$ computations and has

$$\text{Adv}^{kr}_{2\text{DES}}(A_{\text{MinM}}) = 1.$$  

This uses $q = 1$ and is unlikely to return the target key. For that one should extend the attack to a larger value of $q$.  

Block ciphers

$$3DES_3 : \{0, 1\}^{168} \times \{0, 1\}^{64} \rightarrow \{0, 1\}^{64}$$

$$3DES_2 : \{0, 1\}^{112} \times \{0, 1\}^{64} \rightarrow \{0, 1\}^{64}$$

are defined by

$$3DES_3 K_1 \| K_2 \| K_3 (M) = \text{DES}_{K_3} (\text{DES}^{-1}_{K_2} (\text{DES}_{K_1}(M)))$$

$$3DES_2 K_1 \| K_2 (M) = \text{DES}_{K_2} (\text{DES}^{-1}_{K_1} (\text{DES}_{K_2}(M)))$$

Meet-in-the-middle attack on $3DES_3$ reduces its “effective” key length to 112.
DESX

\[ \text{DESX}_{K_1, K_2}(M) = K_2 \oplus \text{DES}_K(K_1 \oplus M) \]

- Key length = 56 + 64 + 64 = 184
- “effective” key length = 120 due to a \(2^{120}\) time meet-in-middle attack
- No more resistant than DES to linear or differential cryptanalysis

Good practical replacement for DES that has lower computational cost than 2DES or 3DES.
Later we will see “birthday” attacks that “break” a block cipher
\[ E : \{0, 1\}^k \times \{0, 1\}^\ell \rightarrow \{0, 1\}^\ell \text{ in time } 2^{\ell/2} \]

For DES this is \(2^{64/2} = 2^{32}\) which is small, and this is unchanged for 2DES and 3DES.

Would like a larger block size.
1998: NIST announces competition for a new block cipher

- key length 128
- block length 128
- faster than DES in software

Submissions from all over the world: MARS, Rijndael, Two-Fish, RC6, Serpent, Loki97, Cast-256, Frog, DFC, Magenta, E2, Crypton, HPC, Safer+, Deal
1998: NIST announces competition for a new block cipher

- key length 128
- block length 128
- faster than DES in software

Submissions from all over the world: MARS, Rijndael, Two-Fish, RC6, Serpent, Loki97, Cast-256, Frog, DFC, Magenta, E2, Crypton, HPC, Safer+, Deal

2001: NIST selects Rijndael to be AES.
function $AES_K(M)$

$(K_0, \ldots, K_{10}) \leftarrow \text{expand}(K)$

$s \leftarrow M \oplus K_0$

for $r = 1$ to $10$ do

$s \leftarrow S(s)$

$s \leftarrow \text{shift-rows}(s)$

if $r \leq 9$ then $s \leftarrow \text{mix-cols}(s)$ fi

$s \leftarrow s \oplus K_r$

end for

return $s$

- Fewer tables than DES
- Finite field operations
The AES movie

http://www.youtube.com/watch?v=H2L1H0w_ANg
Implementing AES

<table>
<thead>
<tr>
<th>Pre-compute and store round function tables</th>
<th>Code size</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-compute and store S-boxes only</td>
<td>smaller</td>
<td>slower</td>
</tr>
<tr>
<td>No pre-computation</td>
<td>smallest</td>
<td>slowest</td>
</tr>
</tbody>
</table>

**AES-NI:** Hardware for AES, now present on most processors. Your laptop may have it! Can run AES at around 1 cycle/byte. VERY fast!
Best known key-recovery attack [BoKhRe11] takes $2^{126.1}$ time, which is only marginally better than the $2^{128}$ time of EKS.

There are attacks on reduced-round versions of AES as well as on its sibling algorithms AES192, AES256. Many of these are “related-key” attacks. There are also effective side-channel attacks on AES such as “cache-timing” attacks [Be05,OsShTr05].
Exercise

Define $F$: $\{0, 1\}^{256} \times \{0, 1\}^{256} \rightarrow \{0, 1\}^{256}$ by

$\textbf{Alg } F_{K_1\parallel K_2}(x_1\parallel x_2)$

$y_1 \leftarrow \text{AES}^{-1}(K_1, x_1 \oplus x_2); \ y_2 \leftarrow \text{AES}(K_2, x_2 \oplus K_1)$

Return $y_1\parallel y_2$

for all 128-bit strings $K_1, K_2, x_1, x_2$. Let $T_{\text{AES}}$ denote the time for one computation of AES or $\text{AES}^{-1}$. Below, running times are worst-case and should be functions of $T_{\text{AES}}$.

1. Prove that $F$ is a blockcipher.

2. What is the running time of a 4-query exhaustive key-search attack on $F$?

3. Give the best 4-query key-recovery attack that you can on $F$ in the form of an adversary $A$ specified in pseudocode and achieving $\text{Adv}^{kr}_F(A) = 1$. Say what is the running time of $A$. 
So far, a block cipher has been viewed as secure if it resists key recovery, meaning there is no efficient adversary $A$ having $\text{Adv}^\text{kr}_E(A) \approx 1$.

Is security against key recovery enough?

Not really. For example define $E$: $\{0,1\}^{128} \times \{0,1\}^{256} \rightarrow \{0,1\}^{256}$ by


This is as secure against key-recovery as AES, but not a “good” blockcipher because half the message is in the clear in the ciphertext.
Possible reaction: But DES, AES are not designed like $E$ above, so why does this matter?

Answer: It tells us that security against key recovery is not, as a block-cipher property, sufficient for security of uses of the block cipher.

As designers and users we want to know what properties of a block cipher give us security when the block cipher is used.
So what is a “good” block cipher?

<table>
<thead>
<tr>
<th>Possible Properties</th>
<th>Necessary?</th>
<th>Sufficient?</th>
</tr>
</thead>
<tbody>
<tr>
<td>security against key recovery</td>
<td>YES</td>
<td>NO!</td>
</tr>
<tr>
<td>hard to find $M$ given $C = E_K(M)$</td>
<td>YES</td>
<td>NO!</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

We can’t define or understand security well via some such (indeterminable) list.

We want a single “master” property of a block cipher that is sufficient to ensure security of common usages of the block cipher.