BLOCK CIPHERS
and
KEY-RECOVERY SECURITY
There are only 10 types of people in the world: Those who understand binary and those who don't.
\{0, 1\}^n \text{ is the set of } n\text{-bit strings and } \{0, 1\}^* \text{ is the set of all strings of finite length. By } \varepsilon \text{ we denote the empty string.}

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

If } x \text{ is a string then } |x| \text{ denotes its length. Example: } |0100| = 4.

By } x \leftarrow S \text{ we denote picking an element at random from set } S \text{ and assigning it to } x. \text{ Thus } \Pr[x = s] = 1/|S| \text{ for every } s \in S.
By $f : \text{Dom} \rightarrow \text{Rng}$ we denote a function taking an input $x$ in the domain Dom and returning an output $f(x)$ in the range Rng.

We say that $f : \text{Dom} \rightarrow \text{Rng}$ is a permutation if Dom = Rng and there is an inverse function $f^{-1} : \text{Rng} \rightarrow \text{Dom}$ satisfying

$$\forall x \in \text{Dom} : f^{-1}(f(x)) = x$$

This means $f$ must be one-to-one and onto: for every $y \in \text{Rng}$ there is a unique $x \in \text{Dom}$ such that $f(x) = y$. 
Consider the following two functions $f : \{0, 1\}^2 \to \{0, 1\}^2$, where $\text{Dom} = \text{Rng} = \{0, 1\}^2$:

\[
\begin{array}{cccc}
 x & 00 & 01 & 10 & 11 \\
 f(x) & 01 & 11 & 00 & 10 \\
\end{array}
\]

A permutation

\[
\begin{array}{cccc}
 x & 00 & 01 & 10 & 11 \\
 f(x) & 01 & 11 & 11 & 10 \\
\end{array}
\]

Not a permutation

\[
\begin{array}{cccc}
 x & 00 & 01 & 10 & 11 \\
 f^{-1}(x) & 10 & 00 & 11 & 01 \\
\end{array}
\]

Its inverse
A family of functions (also called a function family) is a two-argument function $F : \text{Keys} \times \text{Dom} \rightarrow \text{Rng}$. For $K \in \text{Keys}$ we let $F_K : \text{Dom} \rightarrow \text{Rng}$ be defined by

$$\forall x \in \text{Dom} : F_K(x) = F(K, x)$$
Let $E: \text{Keys} \times \text{Dom} \rightarrow \text{Rng}$ be a family of functions. We say that $E$ is a block cipher if

- $\text{Rng} = \text{Dom}$
- $E_K: \text{Dom} \rightarrow \text{Dom}$ is a permutation for every key $K \in \text{Keys}$, meaning has an inverse $E_K^{-1}: \text{Dom} \rightarrow \text{Dom}$
- $E, E^{-1}$ are efficiently computable,

where $E^{-1}: \text{Keys} \times \text{Dom} \rightarrow \text{Dom}$, defined by $E^{-1}(K, y) = E_K^{-1}(y)$, is the inverse block cipher to $E$.

If $\text{Keys} = \{0, 1\}^k$ we call $k$ the key length. If $\text{Dom} = \{0, 1\}^\ell$ we call $\ell$ the block length.
Block ciphers: Example

This table describes a block cipher $E: \{0, 1\}^2 \times \{0, 1\}^2 \rightarrow \{0, 1\}^2$, meaning $\text{Keys} = \text{Dom} = \{0, 1\}^2$. 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 $E^{-1}: \{0, 1\}^2 \times \{0, 1\}^2 \rightarrow \{0, 1\}^2$ is given by the same table: the table entry corresponding to the key in row $K$ and output in column $y$ is $E_K^{-1}(y)$. That is, $E^{-1} = E$. Of course this is not always true.
Let $\ell = k$ and define $E : \{0, 1\}^k \times \{0, 1\}^\ell \rightarrow \{0, 1\}^\ell$ 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 $E: \text{Keys} \times \text{Dom} \rightarrow \text{Dom}$ 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.
Slow is good
Let $E : \text{Keys} \times \text{Dom} \rightarrow \text{Dom}$ be a block cipher. Is $E$ a permutation?

**How to proceed to answer this:** Think slow. Don’t jump to a conclusion. Instead:

- Look back at the definition of a block cipher.
- Look back at the definition of a permutation.
- Pattern match these.
- Now make an informed and justified conclusion.

This is an exercise in correct mathematical language.

This is considered a *high-school level* exercise.
Let $E: \{0, 1\}^k \times \{0, 1\}^\ell \rightarrow \{0, 1\}^\ell$ be a block cipher. It is considered public. In typical usage

- $K \leftarrow \{0, 1\}^k$ is known to parties $S, R$, but not given to adversary $A$.
- $S, R$ use $E_K$ for encryption

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.
FIPS PUB 46-3

FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION

Reaffirmed
1999 October 25

U.S. DEPARTMENT OF COMMERCE/National Institute of Standards and Technology

DATA ENCRYPTION STANDARD (DES)
Key Length $k = 56$

Block length $\ell = 64$

So,

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

$$\text{DES}^{-1}: \{0, 1\}^{56} \times \{0, 1\}^{64} \to \{0, 1\}^{64}$$
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 ≤ i ≤ 16
\(M \leftarrow \text{IP}(M)\)

Parse \(M\) as \(L_0 \parallel R_0\) \hspace{1em} // |L_0| = |R_0| = 32
for \(i = 1\) to 16 do
\(L_i \leftarrow R_{i-1}\); \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 \(C\)

Round \(i\):

Invertible given \(K_i\):
function $\text{DES}_K(M)$ \hspace{1em} // $|K| = 56$ and $|M| = 64$
$(K_1, \ldots, K_{16}) \leftarrow \text{KeySchedule}(K) \hspace{1em} // |K_i| = 48$ for $1 \leq i \leq 16$
$M \leftarrow \text{IP}(M)$
Parse $M$ as $L_0 || R_0$ \hspace{1em} // $|L_0| = |R_0| = 32$
for $i = 1$ to $16$ do
    $L_i \leftarrow R_{i-1}$ ;  \hspace{0.5em} $R_i \leftarrow f(K_i, R_{i-1}) \oplus L_{i-1}$
$C \leftarrow \text{IP}^{-1}(L_{16} || R_{16})$
return $C$

function $\text{DES}_K^{-1}(C)$ \hspace{1em} // $|K| = 56$ and $|M| = 64$
$(K_1, \ldots, K_{16}) \leftarrow \text{KeySchedule}(K) \hspace{1em} // |K_i| = 48$ for $1 \leq i \leq 16$
$C \leftarrow \text{IP}(C)$
Parse $C$ as $L_{16} || R_{16}$
for $i = 16$ down to $1$ do
    $R_{i-1} \leftarrow L_i$ ;  \hspace{0.5em} $L_{i-1} \leftarrow f(K_i, R_{i-1}) \oplus R_i$
$M \leftarrow \text{IP}^{-1}(L_0 || R_0)$
return $M$
**DES Construction**

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

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

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

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

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 $C$

<table>
<thead>
<tr>
<th>IP</th>
<th>IP$^{-1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>58</td>
<td>40</td>
</tr>
<tr>
<td>60</td>
<td>39</td>
</tr>
<tr>
<td>62</td>
<td>38</td>
</tr>
<tr>
<td>64</td>
<td>37</td>
</tr>
<tr>
<td>57</td>
<td>36</td>
</tr>
<tr>
<td>59</td>
<td>35</td>
</tr>
<tr>
<td>61</td>
<td>34</td>
</tr>
<tr>
<td>63</td>
<td>33</td>
</tr>
<tr>
<td>50</td>
<td>48</td>
</tr>
<tr>
<td>52</td>
<td>47</td>
</tr>
<tr>
<td>54</td>
<td>46</td>
</tr>
<tr>
<td>56</td>
<td>45</td>
</tr>
<tr>
<td>42</td>
<td>44</td>
</tr>
<tr>
<td>44</td>
<td>43</td>
</tr>
<tr>
<td>46</td>
<td>42</td>
</tr>
<tr>
<td>48</td>
<td>41</td>
</tr>
<tr>
<td>26</td>
<td>16</td>
</tr>
<tr>
<td>28</td>
<td>15</td>
</tr>
<tr>
<td>30</td>
<td>14</td>
</tr>
<tr>
<td>32</td>
<td>13</td>
</tr>
<tr>
<td>18</td>
<td>12</td>
</tr>
<tr>
<td>20</td>
<td>11</td>
</tr>
<tr>
<td>22</td>
<td>10</td>
</tr>
<tr>
<td>24</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>12</td>
<td>7</td>
</tr>
<tr>
<td>14</td>
<td>6</td>
</tr>
<tr>
<td>16</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
</tr>
</tbody>
</table>
function \( f(J, R) \)  
\[
R \leftarrow E(R) ; \quad R \leftarrow R \oplus J \\
\text{Parse } R \text{ as } R_1 \parallel R_2 \parallel R_3 \parallel R_4 \parallel R_5 \parallel R_6 \parallel R_7 \parallel R_8 \\
\text{for } i = 1, \ldots, 8 \text{ do} \\
\quad R_i \leftarrow S_i(R_i) \quad \text{// Each S-box returns 4 bits} \\
R \leftarrow R_1 \parallel R_2 \parallel R_3 \parallel R_4 \parallel R_5 \parallel R_6 \parallel R_7 \parallel R_8 \\
\text{// } |R| = 32 \text{ bits} \\
R \leftarrow P(R) ; \text{return } R
\]
### 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>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>S_1</strong>:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
</tr>
<tr>
<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>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>0 1</td>
<td>15</td>
<td>7</td>
<td>4</td>
<td>14</td>
<td>2</td>
<td>13</td>
<td>10</td>
<td>6</td>
<td>12</td>
<td>11</td>
<td>9</td>
<td>5</td>
<td>3</td>
<td>8</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<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>
<td>5</td>
<td>0</td>
</tr>
<tr>
<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>
<td>6</td>
<td>13</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>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>S_2</strong>:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
</tr>
<tr>
<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>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>0 1</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>
<td>11</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<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>
<td>2</td>
<td>15</td>
</tr>
<tr>
<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>
<td>14</td>
<td>9</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>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>S_3</strong>:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
</tr>
<tr>
<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>
<td>2</td>
<td>8</td>
</tr>
<tr>
<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>
<td>15</td>
<td>1</td>
</tr>
<tr>
<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>
<td>14</td>
<td>7</td>
</tr>
<tr>
<td>1 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>
<td>2</td>
<td>12</td>
<td>2</td>
</tr>
</tbody>
</table>
Let $E: \text{Keys} \times \text{Dom} \rightarrow \text{Rng}$ be a family of functions. It is known to the adversary $A$.

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

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

- A *target key* $K \leftarrow \$ \text{Keys}$ is selected but not given to $A$.
- $A$ can submit a plaintext $M \in \text{Dom}$ and get back $C = E(K, M)$, in this way gathering input-output examples $(M_1, C_1), \ldots, (M_q, C_q)$ of $E_K$.
- $A$ 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 $K \leftarrow^$ Keys is selected but not given to $A$.
- $A$ can submit a plaintext $M \in \text{Dom}$ 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 $(M_1, C_1), \ldots, (M_q, C_q)$.

For many block ciphers, 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 $E: \text{Keys} \times \text{Dom} \rightarrow \text{Rng}$ be a family of functions, and $A$ an adversary.

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

**procedure Initialize**

$K \leftarrow$ Keys; $i \leftarrow 0$

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

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

$C_i \leftarrow E(K, M_i)$

Return $C_i$

**procedure $\text{Finalize}(K')$**

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

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

- If $E(K', M_j) \neq C_j$ then $\text{win} \leftarrow \text{false}$
- If $M_j \in \{M_1, \ldots, M_{j-1}\}$ then $\text{win} \leftarrow \text{false}$

Return $\text{win}$

\[ \text{Adv}^{\text{kr}}_E(A) = \Pr[\text{KR}_E^A \Rightarrow \text{true}] \]

We say that $A$ is a $q$-query adversary if it makes $q$ (distinct) queries to its $\text{Fn}$ oracle.
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 attack

Let $E: \text{Keys} \times \text{Dom} \rightarrow \text{Rng}$ be a function family with $\text{Keys} = \{T_1, \ldots, T_N\}$ and $\text{Dom} = \{x_1, \ldots, x_d\}$. Let $1 \leq q \leq d$ be a parameter.

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

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

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

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

Then $\text{Adv}^{\text{kr}}_E(A_{\text{eks}}) = 1$ because $K \in \{T_1, \ldots, T_N\}$ and $K$ is consistent with $(M_1, C_1), \ldots, (M_q, C_q)$. 
Exhaustive Key Search attack

Let $E : \text{Keys} \times \text{Dom} \to \text{Rng}$ be a function family with $\text{Keys} = \{ T_1, \ldots, T_N \}$ and $\text{Dom} = \{ x_1, \ldots, x_d \}$. Let $1 \leq q \leq d$ be a parameter.

**adversary $A_{eks}$**

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

For $i = 1, \ldots, N$ 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 $K \in \{ T_1, \ldots, T_N \}$ and $K$ is consistent with $(M_1, C_1), \ldots, (M_q, C_q)$.

**Informal claim:** Think of $q$ as small, like $q \in \{1, 2, 3\}$. Suppose $E : \{0, 1\}^k \times \{0, 1\}^\ell \to \{0, 1\}^\ell$ is a standard block cipher. Then as long as $q > k/\ell$, empirical evidence says that the attack returns the target key $K$ itself.
Exercise: Target key recovery

Let $E: \text{Keys} \times \text{Dom} \rightarrow \text{Rng}$ be a family of functions and $A$ an adversary. The following measures $A$’s ability to find the target key:

<table>
<thead>
<tr>
<th>Game $\text{TKR}_E$</th>
<th><strong>procedure</strong> $\text{Fn}(M)$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>procedure</strong> $\text{Initialize}$</td>
<td>Return $E(K, M)$</td>
</tr>
<tr>
<td>$K \leftarrow \text{Keys}$</td>
<td><strong>procedure</strong> $\text{Finalize}(K')$</td>
</tr>
<tr>
<td>Return $(K = K')$</td>
<td></td>
</tr>
</tbody>
</table>

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

Let $k, \ell \geq 1$ be given integers. Present in pseudocode a blockcipher $E: \{0, 1\}^k \times \{0, 1\}^\ell \rightarrow \{0, 1\}^\ell$ for which you do the following:

1. Given any integer $q \leq 2^\ell$, present in pseudocode a $q$-query adversary $A_q$ with $\text{Adv}^{\text{tkr}}_E(A_q) = 1$.
2. Prove that $\text{Adv}^{\text{tkr}}_E(A) \leq 2^{-k}$ for any adversary $A$. 
How long does exhaustive key search take?

DES can be computed at $1.6 \text{ Gbits/sec}$ 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 $\Rightarrow 22.5 \text{ years}$

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 $A_{\text{eks}}$

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

For $i = 1, \ldots, N$ 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
EKS revisited

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

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

For $i = 1, \ldots, N$ 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 $M_j \leftarrow x_j$; $C_j \leftarrow F_n(M_j)$

For $i = 1, \ldots, N$ 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
RSA DES challenges

\[ K \leftarrow^S \{0, 1\}^{56} ; \ Y \leftarrow \text{DES}(K, X) ; \] Publish \ Y \ on \ website.

Reward for recovering \ X

<table>
<thead>
<tr>
<th>Challenge</th>
<th>Post Date</th>
<th>Reward</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>1997</td>
<td>$10,000</td>
<td>Distributed.Net: 4 months</td>
</tr>
<tr>
<td>II</td>
<td>1998</td>
<td>Depends how fast you find key</td>
<td>Distributed.Net: 41 days. EFF: 56 hours</td>
</tr>
<tr>
<td>III</td>
<td>1998</td>
<td>As above</td>
<td>&lt; 28 hours</td>
</tr>
</tbody>
</table>
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 $K_1K_2$ is a target 2DES key and adversary has $M, C$ such that

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

Then

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

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

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

Table $L$

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

Table $R$

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

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

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

Table $L$

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

Table $R$

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.

\[ \text{adversary } A_{\text{MinM}} \]

\[ M_1 \leftarrow 0^{64}; \quad C_1 \leftarrow \text{Fn}(M_1) \]

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

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

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

\[ 3DES_{K_1 \| K_2 \| K_3}(M) = DES_{K_3}(DES_{K_2}^{-1}(DES_{K_1}(M))) \]
\[ 3DES_{K_1 \| K_2}(M) = DES_{K_2}(DES_{K_1}^{-1}(DES_{K_2}(M))) \]

Meet-in-the-middle attack on \(3DES_3\) reduces its “effective” key length to 112.
Later we will see “birthday” attacks that “break” a block cipher
\( E : \{0, 1\}^k \times \{0, 1\}^\ell \rightarrow \{0, 1\}^\ell \) 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 $\text{AES}_K(M)$

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

$s \leftarrow M \oplus K_0$

for $r = 1$ to $10$ do

\begin{align*}
    s &\leftarrow S(s) \\
    s &\leftarrow \text{shift-rows}(s) \\
    \text{if } r \leq 9 \text{ then } s &\leftarrow \text{mix-cols}(s) \text{ fi}
\end{align*}

$s \leftarrow s \oplus K_r$

end for

return $s$

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

http://www.youtube.com/watch?v=H2LlH0w_ANg
### Implementing AES

**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!

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

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

$$\text{Alg } F_{K_1||K_2}(x_1||x_2)$$

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

Return $y_1||y_2$

for all 128-bit strings $K_1, K_2, x_1, x_2$, where $\overline{x}$ denotes the bitwise complement of $x$. (For example $01 = 10$.) Let $T_{\text{AES}}$ denote the time for one computation of AES or AES$^{-1}$. Below, running times are worst-case and should be functions of $T_{\text{AES}}$. 
Exercise

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 a 4-query key-recovery attack in the form of an adversary $A$ specified in pseudocode, achieving $\text{Adv}^\text{kr}_F(A) = 1$ and having running time $O(2^{128} \cdot T_{\text{AES}})$ where the big-oh hides some small constant.
So far, a block cipher has been viewed as secure if it resists key recovery, meaning there is no efficient adversary $A$ having $\text{Adv}_{E}^{\text{kr}}(A) \approx 1$.

**Is security against key recovery enough?**

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

$$E_K(M[1]M[2]) = M[1] \| \text{AES}_K(M[2])$$

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.