A permutation of a set $X$ is a function $f : X \rightarrow X$ such that for every $y$ there is a unique $x$ with $f(x) = y$

- If $|X| = n$ then there are $n!$ permutations of $X$
- If $f$ is a permutation then there is another permutation $g$ such that $g(f(x)) = x$. $g = f^{-1}$.
- If $f$ and $g$ are permutations, then so is $f \circ g$
Basic Components of Block Encryption

Let $\mathcal{B}_n$ denote the set of bit strings of length $n$.

Thus $|\mathcal{B}_n| = 2^n$.

Suppose plaintext and cyphertext are arbitrary bit strings.

Want an encryption algorithm $E$ and a keyspace $\mathcal{K}$ so that $E_K$ is a permutation of $\mathcal{B}_n$, for each $K \in \mathcal{K}$. 
To encrypt, divide plaintext into \textbf{blocks} of length $n$, and apply $E_K$ to each block.

\begin{itemize}
  \item \textbf{ECB mode} (Electronic Code Book mode)
\end{itemize}

How large should $n$ be? (i.e. the block size)

How long should $K$ be? (i.e. the key length)
If \( n \) is small, it would take only a small amount of known-plaintext to compile a “dictionary” or a frequency distribution.

If \( n \) is too large, encryption becomes unwieldy.

Current philosophy: \( n \geq 128 \).
What should the key look like?

Key space $\mathcal{K}$ should be large enough to prevent a

**brute-force search:**
Attacker tries every possible key in $\mathcal{K}$ until one of them works.

Most natural as a known-plaintext attack, but can also be used in a ciphertext-only attack.

If each key requires $k$ bits, then $|\mathcal{K}| \leq 2^k$. 

© 2011 Clifford Bergman
Unfortunately, it is not possible to use the whole “substitution table” $E_K$ for the key.
We would like to design an algorithm that takes as input a key of a reasonable number of bits (100? 1000?) and yields as output a permutation of $B_n$ that looks “random”.

- Each output bit should depend on every input bit and every key bit.
- Changing one input bit should cause approximately half of the output bits to change.
Idea: Build $E_K$ as a composite of “simpler” permutations, some of which depend on $K$. 
Linear and affine transformations

Let $M$ be an $m \times n$ Boolean matrix (using $\oplus$ and $\cdot$ modulo 2). Let $a \in \mathcal{B}_m$ be fixed. We have mappings $\mathcal{B}_n \rightarrow \mathcal{B}_m$ given by

\[
\begin{align*}
\mathbf{b} & \mapsto M\mathbf{b} \quad \text{linear} \\
\mathbf{b} & \mapsto M\mathbf{b} \oplus a \quad \text{affine}
\end{align*}
\]

These are permutations of $\mathcal{B}_n$ iff $n = m$ and $\det M \neq 0$. 
\[ M = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} \]

\[ E(b) = Mb \]
\[ E(b) = \begin{pmatrix}
1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\
1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\
1 & 1 & 0 & 0 & 1 & 0 & 1 & 1
\end{pmatrix} \cdot \begin{pmatrix}
0 \\
1 \\
0 \\
1 \\
1 \\
1 \\
0 \\
0
\end{pmatrix} = \begin{pmatrix}
1 \\
1 \\
0 \\
1 \\
0 \\
0 \\
0 \\
0
\end{pmatrix} \]
Projections

Let $\sigma: \{1, 2, \ldots, m\} \to \{1, 2, \ldots, n\}$ be any function.

Define $\hat{\sigma}: \mathcal{B}_n \to \mathcal{B}_m$ by

$$\hat{\sigma}(b_1 b_2 \ldots, b_n) = b_{\sigma_1} b_{\sigma_2} \ldots b_{\sigma_m}.$$ 

Not necessarily a permutation. Always linear.

Shorthand: write $\langle \sigma(1), \sigma(2), \ldots, \sigma(m) \rangle$ to denote $\sigma$. 

© 2011 Clifford Bergman
Example

\( \sigma: \{1, 2, \ldots, 6\} \to \{1, 2, \ldots, 8\} \)
\( \sigma = \langle 3, 8, 1, 5, 1, 2 \rangle \)

\( b = 01011000 \) then \( \hat{\sigma}(b) = 000101 \)

\( \tau: \{1, \ldots, 8\} \to \{1, \ldots, 8\} \)
\( \tau = \langle 6, 2, 8, 1, 3, 5, 4, 7 \rangle \)

\( \hat{\tau}(b) = 01000110 \)
Small-block substitution

Let $f : \mathcal{B}_n \rightarrow \mathcal{B}_m$ be an arbitrary function.

The table for $f$ can be stored in $2^n m$ bits.

Not practical unless $n$ is small.

$n \times m$ S-box

Obviously, not necessarily a permutation.
Example: Baby Lucifer

Blocksize: 8 bits

Key: 4 bits

Ingredients: two $4 \times 4$ S-boxes $f_0, f_1$, each a permutation of $\mathcal{B}_4$ and a permutation, $\sigma$, of \{1, 2, \ldots, 8\}.
Encryption algorithm:

**input block:** \( b = \underbrace{b_1 b_2 \ldots b_8}_8 \)

**key:** \( k_1 k_2 k_3 k_4 \)

1. Split \( b \) into two halves: \( b_1 b_2 b_3 b_4 \quad b_5 b_6 b_7 b_8 \)
2. Apply \( f_{k_1} \) to first half; \( f_{k_2} \) to second half:
   \[
   b'_1 b'_2 b'_3 b'_4 = f_{k_1}(b_1 b_2 b_3 b_4) \\
   b'_5 b'_6 b'_7 b'_8 = f_{k_2}(b_5 b_6 b_7 b_8)
   \]
3. Merge the output and apply \( \hat{\sigma} \) to the result.
4. Repeat steps 1–3 a second time using \( k_3 \) and \( k_4 \)
\[ \sigma = \langle 7, 8, 5, 2, 1, 3, 6, 4 \rangle \]

\[ \sigma^{-1} = \langle 5, 4, 6, 8, 3, 7, 1, 2 \rangle \]

Thus, \( \sigma(1) = 7, \quad \sigma(2) = 8, \ldots, \sigma(8) = 4 \)
<table>
<thead>
<tr>
<th>$b$</th>
<th>$f_0(b)$</th>
<th>$f_1(b)$</th>
<th>$b$</th>
<th>$f_0(b)$</th>
<th>$f_1(b)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>1101</td>
<td>0001</td>
<td>1000</td>
<td>1110</td>
<td>1010</td>
</tr>
<tr>
<td>0001</td>
<td>0000</td>
<td>0100</td>
<td>1001</td>
<td>0011</td>
<td>1111</td>
</tr>
<tr>
<td>0010</td>
<td>1011</td>
<td>1011</td>
<td>1010</td>
<td>0101</td>
<td>0110</td>
</tr>
<tr>
<td>0011</td>
<td>0111</td>
<td>1101</td>
<td>1011</td>
<td>1100</td>
<td>1000</td>
</tr>
<tr>
<td>0100</td>
<td>0100</td>
<td>1100</td>
<td>1100</td>
<td>0010</td>
<td>0000</td>
</tr>
<tr>
<td>0101</td>
<td>1001</td>
<td>0011</td>
<td>1101</td>
<td>1111</td>
<td>0101</td>
</tr>
<tr>
<td>0110</td>
<td>0001</td>
<td>0111</td>
<td>1110</td>
<td>1000</td>
<td>1001</td>
</tr>
<tr>
<td>0111</td>
<td>1010</td>
<td>1110</td>
<td>1111</td>
<td>0110</td>
<td>0010</td>
</tr>
</tbody>
</table>
Example: \( b = 01110010 \quad k = 0 \ 1 \ 1 \ 1 \)

\[
b = 01110010 \xrightarrow{(f_0,f_1)} 10101011 \xrightarrow{\hat{\sigma}} 11101100
\]

\[
11101100 \xrightarrow{(f_1,f_1)} 10010000 \xrightarrow{\hat{\sigma}} 00001001 = c
\]