Block Ciphers

Chester Rebeiro
IIT Madras

STINSON : chapters 3
Block Cipher

Encryption key is the same as the decryption key ($K_E = K_D$)
Block Cipher : Encryption

- A block cipher encryption algorithm encrypts $n$ bits of plaintext at a time
- May need to pad the plaintext if necessary
- $y = e_k(x)$
A block cipher decryption algorithm recovers the plaintext from the ciphertext.

$x = d_k(y)$
Inside the Block Cipher
(an iterative cipher)

- Each round has the same endomorphic cryptosystem, which takes a key and produces an intermediate output.
- Size of the key is huge... much larger than the block size.
Inside the Block Cipher
(the key schedule)

- A single secret key of fixed size used to generate ‘round keys’ for each round
Inside the Round Function

- **Add Round key:**
  Mixing operation between the round input and the round key.
  Typically, an ex-or operation.

- **Confusion layer:**
  Makes the relationship between round input and output complex.
  An attacker cannot determine the round key even after knowing large number of input-output pairs.

- **Diffusion layer:**
  Dissipate the round input.
  *Avalanche effect:* A single bit change in the round input should cause huge changes in the output.
  Makes it difficult for the attacker to pick out some bits over the others (think Hill cipher)
Achieving Confusion and Diffusion (Substitution-Permutation Networks)

- Confusion achieved by small substitution functions
- Diffusion achieved by diffusion functions
  - Permutations
  - Linear Transformations
Diffusion with Permutations

- Spreads the output of one s-box to other s-boxes
- Thus causing a diffusion.
  - A single bit change in one input (before S1 for instance) affects four inputs of the next round
- Bit wise permutations efficient in hardware but not in software implementations
Permutation Layer Types

- **straight (24x24)**

  0th bit of input goes to 1st bit of output
  1st bit of input goes to 15th bit of output

<table>
<thead>
<tr>
<th>01</th>
<th>15</th>
<th>02</th>
<th>13</th>
<th>06</th>
<th>17</th>
<th>03</th>
<th>19</th>
<th>09</th>
<th>04</th>
<th>21</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>05</td>
<td>12</td>
<td>16</td>
<td>18</td>
<td>07</td>
<td>24</td>
<td>10</td>
<td>23</td>
<td>08</td>
<td>22</td>
<td>20</td>
</tr>
</tbody>
</table>

- **expansion (12x24)**

<table>
<thead>
<tr>
<th>01</th>
<th>03</th>
<th>02</th>
<th>01</th>
<th>06</th>
<th>17</th>
<th>03</th>
<th>07</th>
<th>09</th>
<th>04</th>
<th>09</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>02</td>
<td>05</td>
<td>12</td>
<td>04</td>
<td>06</td>
<td>07</td>
<td>12</td>
<td>10</td>
<td>11</td>
<td>08</td>
<td>10</td>
<td>08</td>
</tr>
</tbody>
</table>

- **compression (24x12)**

  | 01 | 15 | 02 | 13 | 06 | 17 | 03 | 19 | 09 | 04 | 21 | 11 |
Permutation Layer (more variants)

- Common permutation operations which are used in block ciphers
  - circular shift
    - Circular shift input N bits to right (or left)
  - swap
    - Special case of circular shift with shift = N/2
Diffusion with Linear Transformation

• Linear combination of the inputs (can be done byte wise; more software friendly, as no bit manipulations needed)

\[
\begin{bmatrix}
2 & 3 & 1 & 1 \\
1 & 2 & 3 & 1 \\
1 & 1 & 2 & 3 \\
3 & 1 & 1 & 2 \\
\end{bmatrix}
\times
\begin{bmatrix}
x_1 \\
x_2 \\
x_3 \\
x_4 \\
\end{bmatrix}
=
\begin{bmatrix}
y_1 \\
y_2 \\
y_3 \\
y_4 \\
\end{bmatrix}
\text{Example. The AES mix column operation}
\]

• How to choose the linear transformation in the Permutation layer?
  – Need to have good diffusion properties
  – Should have Maximum Branch Number

\[
BranchNumber = \text{MIN}_{a \neq 0} (W(a) + W(F(a)))
\]
Branch Number

\[ \text{BranchNumber} = \min_{a \neq 0} (W(a) + W(F(a))) \]

- **Byte Vector**: Number of non-zero input bytes
- **W(a)**: Byte vector of input (i.e. non-zero bytes in a)
- **W(F(a))**: Byte vector of output (i.e. non-zero bytes in the output)

\[
\begin{bmatrix}
2 & 3 & 1 & 1 \\
1 & 2 & 3 & 1 \\
1 & 1 & 2 & 3 \\
3 & 1 & 1 & 2
\end{bmatrix} \times
\begin{bmatrix}
x_1 \\
x_2 \\
x_3 \\
x_4
\end{bmatrix} =
\begin{bmatrix}
y_1 \\
y_2 \\
y_3 \\
y_4
\end{bmatrix}
\]

**Example**: The AES mix column operation

- Example: AES mix column matrix has a branch number of 5
  - 1 non-zero byte in input causes all 4 bytes of output to change
  - 2 non-zero byte in input causes at-least 3 bytes of output to change (and so on...)
Substitution Layer
(Sbox)

• A lot of the block cipher’s security rests with this.
• Replaces its input with another

<table>
<thead>
<tr>
<th>$z$</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>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\pi_s(z)$</td>
<td>E</td>
<td>4</td>
<td>D</td>
<td>1</td>
<td>2</td>
<td>F</td>
<td>B</td>
<td>8</td>
<td>3</td>
<td>A</td>
<td>6</td>
<td>C</td>
<td>5</td>
<td>9</td>
<td>0</td>
<td>7</td>
</tr>
</tbody>
</table>

• As with the permutation layer, can be straight sbox (mxm)
  expansion sbox (mxn, m<n)
  compression sbox (mxn, m>n)
Sboxes

- In an s-box each output bit can be represented as a function of its input bits

\[
y_1 = f_1(x_1, x_2, x_3, \ldots, x_m)
\]
\[
y_2 = f_2(x_1, x_2, x_3, \ldots, x_m)
\]
\[
y_3 = f_3(x_1, x_2, x_3, \ldots, x_m)
\]
\[
\vdots
\]
\[
y_n = f_n(x_1, x_2, x_3, \ldots, x_m)
\]

The functions have to be non-linear. Linear functions are easily reversed.
S-boxes are Non-linear transformations

\[ y_1 = a_1 x_1 \oplus a_2 x_2 \oplus a_3 x_3 \oplus \cdots \oplus a_m x_m \]
\[ y_2 = a_1 x_1 \oplus a_2 x_2 \oplus a_3 x_3 \oplus \cdots \oplus a_m x_m \]
\[ \cdots = \cdots \]
\[ y_n = a_1 x_1 \oplus a_2 x_2 \oplus a_3 x_3 \oplus \cdots \oplus a_m x_m \]

where \( a_1, a_2, \cdots a_m \in \{0, 1\} \)

- Non-linear s-boxes **do not** have equations like the above.
- Instead they non-linear equations as follows

\[ y_1 = a_1 x_1 \oplus a_2 x_2 x_1 \oplus a_3 x_1 x_5 x_2 \cdots \]
example: Simplified DES SBox

\[ y = S0(x) \]
\[ q || r = S0[a || d][b || c] \]

Non-linear equations for S0

\[ q = abcd + ab + ac + b + d \]
\[ r = abcd + abd + ab + ac + ad + a + c + 1 \]

http://mercury.webster.edu/aleshunas/COSC%205130/G-SDES.pdf
Why Non-linearity?

• We want to make it difficult for reversing an s-box: i.e. determine x from y
  
  – Solving linear equations can be done in polynomial time
  – Solving non-linear equation is NP hard

• Note the difference with the permutation layer, which is a linear layer. The main purpose of the permutation layer is to provide diffusion and not to confuse!
ex-or (An Important Operation)

• Used considerably for key addition

  ▶ Ex-or ($\oplus$) is a binary operation, which results in 1 when both inputs have a different value. Otherwise 0
  ▶ Application of Ex-or in ciphers
    ▶ During Encryption: $p \oplus k = c$
    ▶ During decryption: $c \oplus k = p$

  The same operation can be used during encryption as well as decryption
Block Cipher Design Techniques

- Substitution-Permutation Networks (SPN)
  - AES, PRESENT, SHARK
- Feistel Ciphers
  - DES, CLEFIA, SERPENT, RC5, ... and many more
A Four Round SPN Block Cipher

- An SPN block cipher contains repeating rounds of
  - Key addition
    - Add randomization
  - Substitution
    - A non-linear layer
  - Diffusion
    - A linear layer for spreading

- The repeating randomization, non-linear and linear layers makes it difficult to cryptanalyse

- Used in ciphers such as
  - AES (Advanced Encryption Standard)
  - PRESENT (The Light weight block cipher standard)

SPN: Substitution Permutation Network
**Diffusion in the SPN**

- A single bit of plaintext gets diffused to all bits of the ciphertext.
- If a single bit in the plaintext is flipped
  - Each bit of the ciphertext will flip with probability $1/2$
  - In other words, half the bits of the ciphertext will flip.
- If, even a single bit of the key is wrong, half the bits of the ciphertext is flipped
Decryption

• Is the reverse process
  – Start with the ciphertext and do all operations in the reverse order
  – The round keys are applied in the reverse order
  – Permutation layer should be inverse
  – Substitution (S-boxes) should be inverse
    • This also means that the inverse of the s-box should exist
Feistel Ciphers

- A popular technique for designing block ciphers
  - Examples: DES, RC5, CLEFIA,
- Does not require invertible substitution and permutation layers

Encryption

\[ L_i = R_i \]
\[ R_i = L_i \oplus F(R_{i-1}, K_{i-1}) \]

Decryption

\[ R_{i-1} = L_i \]
\[ L_{i-1} = R_i \oplus F(L_{i-1}, K_{i-1}) \]
What does F contain?

- contains: key mixing, substitution, permutation
- A single round of DES

the sboxes (S1 to S8) are 6x4... they are not invertible
3 round Fiestel cipher

- Iterative
Linear Cryptanalysis
Non-linearity in S-boxes

• In the 1970s, cryptographers took a lot of care in designing s-boxes
  – each output bit of the s-box was the output of a complex non-linear function of the input bits. Like this
    \[ y_1 = a_1 x_1 \oplus a_2 x_1 x_2 \oplus a_3 x_1 x_5 x_2 \cdots \]
  – also, the value of each output bit was un-biased
    i.e. \( \Pr[y_i = 0] = \Pr[y_i = 1] = \frac{1}{2} \) for \( 1 \leq i \leq n \)

This meant that it was difficult to infer anything about \( x \) from an output bit

However....
Linear Approximations

• they overlooked about linear combinations of the s-box output which turned out to be biased...such as

\[ \Pr[y_1 \oplus x_1 \oplus x_5 \oplus x_7 = 0] \ll \frac{1}{2} \quad \text{or} \quad \Pr[y_1 \oplus x_1 \oplus x_5 \oplus x_7 = 1] \gg \frac{1}{2} \]

- low probability of occurrence
- high probability of occurrence

• This bias was exploited by Mitsuru Matsui in 1993 to attack DES. The attack was known as linear cryptanalysis
  - it is a known plaintext attack
  - required $2^{43}$ known plaintext-ciphertext pairs to break DES

background needed for the understanding the attack...
Bias

(A measure of deviation from uniform randomness)

• Consider discrete independent random variables over \{0,1\}

• Let \( \Pr[X_i = 0] = p_i \), thus \( \Pr[X_i = 1] = 1 - p_i \) for \( i=1,2,3,\ldots \).

• Due to independence, the joint probability is obtained by simply multiplying. Thus for \( i \neq j \),

\[
\begin{align*}
\Pr[X_i = 0, X_j = 0] &= p_ip_j, \\
\Pr[X_i = 0, X_j = 1] &= p_i(1 - p_j), \\
\Pr[X_i = 1, X_j = 0] &= (1 - p_i)p_j, \\
\Pr[X_i = 1, X_j = 1] &= (1 - p_i)(1 - p_j).
\end{align*}
\]

• Consider discrete random variables \( X_i \oplus X_j \) where \( i \neq j \)

\[
\begin{align*}
\Pr[X_i \oplus X_j = 0] &= p_ip_j + (1 - p_i)(1 - p_j), \\
\Pr[X_i \oplus X_j = 1] &= p_i(1 - p_j) + (1 - p_i)p_j.
\end{align*}
\]
Bias

• Define **bias** of $X_i$ as $\epsilon_i = p_i - \frac{1}{2}$.

• Some properties of the bias

1. $-\frac{1}{2} \leq \epsilon_i \leq \frac{1}{2}$

2. $\Pr[X_i = 0] = \frac{1}{2} + \epsilon_i$

3. $\Pr[X_i = 1] = \frac{1}{2} - \epsilon_i$

$\Pr[X_i \oplus X_j = 0] = \Pr[X_i = 0] \Pr[X_j = 0] + \Pr[X_i = 1] \Pr[X_j = 1]$

$= \left( \frac{1}{2} + \epsilon_i \right) \left( \frac{1}{2} + \epsilon_j \right) + \left( \frac{1}{2} - \epsilon_i \right) \left( \frac{1}{2} - \epsilon_j \right) = \left( \frac{1}{2} + 2\epsilon_i \epsilon_j \right)$

• If the bias is 0 then $X_i$ can take values of 0 or 1 with equal probability
  The further the bias is from 0 (ie. close to ±1/2) then $X_i$ takes 0 with higher (or lower) probability
• The bias is therefore a measure of the randomness
Linear Approximations of an s-box

How to construct?

Represent the s-box in binary as in the following table.
Linear Approximations of an s-box

Consider a linear combination of inputs and outputs

For example $X_1 \oplus X_4 \oplus Y_2$ and fill in the truth table

\[
\begin{array}{ccccccccc}
X_1 & X_2 & X_3 & X_4 & Y_1 & Y_2 & Y_3 & Y_4 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\
0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\
1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\
\end{array}
\]

$\#1s = 8$

$\#0s = 8$

$p = \Pr[X_1 \oplus X_4 \oplus Y_2 = 0] = 1/2$

$\epsilon = p - \frac{1}{2} = 0$

unbiased

unbiased
Linear Approximations of an s-box

Consider a linear combination of inputs and outputs for example \( X_1 \oplus X_2 \oplus X_3 \oplus Y_2 \) and fill in the truth table.

<table>
<thead>
<tr>
<th>( X_1 )</th>
<th>( X_2 )</th>
<th>( X_3 )</th>
<th>( X_4 )</th>
<th>( Y_1 )</th>
<th>( Y_2 )</th>
<th>( Y_3 )</th>
<th>( Y_4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0</td>
<td>1 1 1 0</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0 0 1</td>
<td>0 1 0 0</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0 1 0</td>
<td>1 1 0 1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0 1 1</td>
<td>0 0 0 1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 1 0 0</td>
<td>0 0 1 0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 1 0 1</td>
<td>1 1 1 1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 1 1 0</td>
<td>1 0 1 1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 1 1 1</td>
<td>1 1 0 0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 0 0 0</td>
<td>0 0 1 1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 0 0 1</td>
<td>1 0 1 0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 0 1 0</td>
<td>0 1 1 0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 0 1 1</td>
<td>1 1 0 0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 1 0 0</td>
<td>0 1 0 1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 1 0 1</td>
<td>1 0 0 1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 1 1 0</td>
<td>0 0 0 0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 1 1 1</td>
<td>0 1 1 1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\( \#1s = 10 \)
\( \#0s = 6 \)

\[ p = \Pr[X_1 \oplus X_2 \oplus X_3 \oplus Y_2 = 0] = \frac{3}{8} \]

\[ \varepsilon = p - \frac{1}{2} = -\frac{1}{8} = -0.125 \]

biased
Linear Approximations of an s-box

Consider another example $X_3 \oplus X_4 \oplus Y_1 \oplus Y_4$ and fill in the truth table

<table>
<thead>
<tr>
<th>X_1</th>
<th>X_2</th>
<th>X_3</th>
<th>X_4</th>
<th>Y_1</th>
<th>Y_2</th>
<th>Y_3</th>
<th>Y_4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

$\#1s = 14$

$\#0s = 2$

$$p = \Pr[X_3 \oplus X_4 \oplus Y_1 \oplus Y_4 = 0] = \frac{1}{8}$$

$$\varepsilon = p - \frac{1}{2} = -\frac{3}{8} = -.375$$

Highly biased
Linear Approximation Tables

\[ \left( \bigoplus_{i=1}^{4} a_i X_i \right) \oplus \left( \bigoplus_{i=1}^{4} b_i Y_i \right) \]

where \( a_i \in \{0, 1\}, b_i \in \{0, 1\}, i = 1, 2, 3, 4. \)

\[ \varepsilon(a, b) = \frac{NL(a, b) - 8}{16} \]

| \( b \) | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 16 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 |
| 1 | 8 | 8 | 6 | 6 | 8 | 8 | 6 | 14 | 10 | 10 | 8 | 8 | 10 | 10 | 8 | 8 |
| 2 | 8 | 8 | 6 | 6 | 8 | 8 | 6 | 10 | 10 | 8 | 8 | 10 | 10 | 8 | 8 |
| 3 | 8 | 8 | 6 | 6 | 8 | 8 | 8 | 10 | 2 | 6 | 6 | 6 | 10 | 10 | 6 | 6 |
| 4 | 8 | 10 | 8 | 6 | 6 | 4 | 6 | 8 | 8 | 6 | 8 | 10 | 10 | 4 | 10 | 8 |
| 5 | 8 | 6 | 6 | 6 | 8 | 6 | 8 | 12 | 10 | 6 | 8 | 4 | 10 | 8 | 6 | 6 |
| 6 | 8 | 10 | 6 | 12 | 10 | 8 | 8 | 10 | 8 | 6 | 10 | 12 | 6 | 8 | 8 | 6 |
| 7 | 8 | 6 | 8 | 10 | 10 | 4 | 10 | 8 | 6 | 8 | 10 | 8 | 12 | 10 | 8 | 10 |
| 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 6 | 10 | 10 | 6 | 10 | 6 | 6 | 2 |
| 9 | 8 | 8 | 6 | 6 | 8 | 8 | 6 | 4 | 8 | 6 | 10 | 8 | 12 | 10 | 8 | 6 |
| A | 8 | 12 | 6 | 10 | 4 | 8 | 10 | 6 | 10 | 10 | 8 | 8 | 10 | 10 | 8 | 8 |
| B | 8 | 12 | 8 | 4 | 12 | 8 | 12 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 |
| C | 8 | 6 | 12 | 6 | 6 | 8 | 10 | 8 | 10 | 8 | 10 | 12 | 8 | 10 | 8 | 6 |
| D | 8 | 10 | 10 | 8 | 6 | 12 | 8 | 10 | 4 | 6 | 10 | 8 | 10 | 8 | 8 | 10 |
| E | 8 | 10 | 10 | 8 | 6 | 4 | 8 | 10 | 6 | 8 | 8 | 6 | 4 | 10 | 6 | 8 |
| F | 8 | 6 | 4 | 6 | 6 | 8 | 10 | 8 | 8 | 6 | 12 | 6 | 6 | 8 | 10 | 8 |

Linear Approximation Table
(captures number of 0s in the truth table)
What does the linear approximations mean

\[ X_3 \oplus X_4 \oplus Y_1 \oplus Y_4 \]

• If we do the following

\[
\text{while(large number of times)} \{
\text{generate a random plaintext}
\text{ } z = \text{ex-or}(x_3, x_4, y_1, y_4)
\}
\]

• The probability that z takes the value 0 is 1/8

How do we use this fact to attack the block cipher?
Piling-up Lemma

Consider two linear combinations of random variables

\[ X_A = X_1 \oplus X_2 \oplus X_3 \quad \text{having bias } \varepsilon_A \]
\[ X_B = X_4 \oplus X_5 \oplus X_6 \quad \text{having bias } \varepsilon_B \]

What is the bias of \( X_A \oplus X_B \)?

The resultant bias \( \varepsilon_{AB} \) can be computed by the Pilingup Lemma

**Lemma (Piling-up lemma)** Let \( \varepsilon_{i_1,i_2,\ldots,i_k} \) denote the bias of the random variable \( X_{i_1} \oplus \cdots \oplus X_{i_k} \). Then

\[
\varepsilon_{i_1,i_2,\ldots,i_k} = 2^{k-1} \prod_{j=1}^{k} \varepsilon_{i_j}.
\]

Proof by Mathematical Induction
The General Attack Scheme

1. Use piling up lemma to identify linear trails in the cipher, which have high bias.
   – Compute the bias till the pen-ultimate round

2. To determine \( k = (K_{5,5} \ldots K_{5,8}) \) do the following
   a. Guess the value of \( k \) (16 possibilities)
   b. Compute \( S^{-1}(k \wedge c_i) \) for each ciphertext (we get a distribution)
   c. Determine if the bias matches the theoretical estimates.
Applying Piling-up Lemma for the cipher

Find paths which are highly biased

\[ T_1 = U_5^1 \oplus U_7^1 \oplus U_8^1 \oplus V_6^1 \text{ has bias } 1/4 \]
\[ T_2 = U_6^2 \oplus V_6^2 \oplus V_8^2 \text{ has bias } -1/4 \]
\[ T_3 = U_6^3 \oplus V_6^3 \oplus V_8^3 \text{ has bias } -1/4 \]
\[ T_4 = U_{14}^3 \oplus V_{14}^3 \oplus V_{16}^3 \text{ has bias } -1/4 \]

\[ T_1 \oplus T_2 \oplus T_3 \oplus T_4 \]
\[ \text{has bias equal to } 2^3(1/4)(-1/4)^3 = -1/32. \]
\[ T_1 = U_5^1 \oplus U_7^1 \oplus U_8^1 \oplus V_6^1 = X_5 \oplus K_5^1 \oplus X_7 \oplus K_7^1 \oplus X_8 \oplus K_8^1 \oplus V_6^1 \]
\[ T_2 = U_6^2 \oplus V_6^2 \oplus V_8^2 = V_6^1 \oplus K_6^2 \oplus V_6^2 \oplus V_8^2 \]
\[ T_3 = U_6^3 \oplus V_6^3 \oplus V_8^3 = V_6^2 \oplus K_6^3 \oplus V_6^3 \oplus V_8^3 \]
\[ T_4 = U_{14}^3 \oplus V_{14}^3 \oplus V_{16}^3 = V_8^2 \oplus K_{14}^3 \oplus V_{14}^3 \oplus V_{16}^3. \]

\[ T_1 \oplus T_2 \oplus T_3 \oplus T_4 \]
\[ X_5 \oplus X_7 \oplus X_8 \oplus V_6^1 \oplus V_6^2 \oplus V_6^3 \oplus V_{14} \oplus V_{16} \]
\[ \oplus K_5^1 \oplus K_7^1 \oplus K_8^1 \oplus K_6^2 \oplus K_6^3 \oplus K_{14}^3 \]

has bias equal to \(2^3(1/4)(-1/4)^3 = -1/32.\)
From the cipher
\[ V_6^3 = U_6^4 \oplus K_8^4 \]
\[ V_8^3 = U_{14}^4 \oplus K_{14}^4 \]
\[ V_{14}^3 = U_8^4 \oplus K_8^4 \]
\[ V_{16}^3 = U_{16}^4 \oplus K_{16}^4 \]

Thus,
\[ X_5 \oplus X_7 \oplus X_8 \oplus V_6^3 \oplus V_8^3 \oplus V_{14}^3 \oplus V_{16}^3 \oplus K_5^1 \oplus K_7^1 \oplus K_8^1 \oplus K_6^2 \oplus K_8^2 \oplus K_1^3 \]

has bias equal to \( 2^3 (1/4)(-1/4)^3 = -1/32 \).

Now, the key part is a constant (either 0 or 1)
\[ K_5^1 \oplus K_7^1 \oplus K_8^1 \oplus K_6^2 \oplus K_8^2 \oplus K_1^3 \]

Thus, bias of \( X_5 \oplus X_7 \oplus X_8 \oplus U_6^4 \oplus U_8^4 \oplus U_{14}^4 \oplus U_{16}^4 \)
is either +1/32 or -1/32 depending on the key bits
The Linear Cryptanalysis Attack

• The attacker needs
  – A large number of plaintext-ciphertext pairs
    • We denote each pair by (x, y) – x: plaintext, y: ciphertext
    • For the Toy cipher above (approx 8000)
    • For a cipher like DES $2^{48}$
  – all plaintexts are encrypted with the same key

• The attack
  1. Guess $k^5_{<2>}$ and $k^5_{<4>}$ (256 possibilities)
  2. For each $y^5_{<2>}$ and $y^5_{<4>}$ compute $v^4_{<2>}$ and $v^4_{<4>}$
  3. Then compute $\text{inv-sbox}(v^4_{<2>})$ and $\text{inv-sbox}(v^4_{<4>})$
    to obtain $u^4_{<2>}$ and $u^4_{<4>}$
  4. Now compute
     \[ z \leftarrow x^5_5 \oplus x^5_7 \oplus x^5_8 \oplus u^4_6 \oplus u^4_8 \oplus u^4_{14} \oplus u^4_{16} \]

If the key guess is correct, the bias of $z$ must be $\pm 1/32$
(i.e. $z$ must be 0 (or 1) with probability $1/2 \pm 1/32$)
If the key guess is wrong, the bias of $z$ must be 0
(i.e. $z$ must be 0 (or 1) with probability 1/2)
The Linear Cryptanalysis Attack

Algorithm \text{LINEAR ATTACK}(\mathcal{T}, T, \pi_s^{-1})

\begin{align*}
\text{for } (L_1, L_2) &\leftarrow (0, 0) \text{ to } (F, F) \\
do & \text{ Count}[L_1, L_2] \leftarrow 0 \\
\text{for each } (x, y) &\in \mathcal{T} \\
& \begin{align*}
& \text{for } (L_1, L_2) \leftarrow (0, 0) \text{ to } (F, F) \\
& \quad \begin{align*}
& \begin{cases} 
& u_4^{<2>} \leftarrow L_1 \oplus y^{<2>} \\
& v_4^{<4>} \leftarrow L_2 \oplus y^{<4>} \\
& v_4^{<2>} \leftarrow \pi_s^{-1}(v_4^{<2>}) \\
& u_4^{<4>} \leftarrow \pi_s^{-1}(v_4^{<4>}) 
\end{cases} \\
& \text{if } z = 0 \\
& \quad \text{then } \text{ Count}[L_1, L_2] \leftarrow \text{ Count}[L_1, L_2] + 1 \\
\end{align*}
\end{align*}

\begin{align*}
& \text{max} \leftarrow -1 \\
& \text{for } (L_1, L_2) \leftarrow (0, 0) \text{ to } (F, F) \\
& \quad \begin{align*}
& \quad \text{ Count}[L_1, L_2] \leftarrow \| \text{ Count}[L_1, L_2] - T/2 \| \\
& \quad \text{if } \text{ Count}[L_1, L_2] > \text{ max} \\
& \quad \quad \text{then } \begin{cases} 
& \text{ max } \leftarrow \text{ Count}[L_1, L_2] \\
& \text{ maxkey } \leftarrow (L_1, L_2) 
\end{cases} \\
\end{align*}
\end{align*}

\text{output } (\text{ maxkey })

This is the guessed key which varies from 0 to 255.

For a key guess, Count counts how often \( z=0 \). For the correct key guess, count should be highest.

For each plaintext-ciphertext pair

Compute \( u_4^{<2>} \) and \( u_4^{<4>} \)

Increment count if \( z=0 \)

Determine most probable key byte of the 256 possible keys
The correct key should have max count value
Wrong keys should have count value approximately \( T/2 \)
Differential Cryptanalysis
Differential Cryptanalysis

• Attributed to Eli Biham and Adi Shamir in CRYPTO’90
  – Although, the idea was known in the 1970s by IBM (and the NSA)
    • In IBM, this used to be known as T-attack or Tickle attack

• Differential cryptanalysis is a chosen plaintext attack
  – It requires $2^{47}$ chosen plaintexts to break DES
Differentials

- If we have two Boolean linear equations such as
  \[ A = a \oplus b \oplus k_1 \oplus k_2 \quad \text{and} \quad B = c \oplus d \oplus k_1 \oplus k_2 \]
- Then, the differential is their ex-or
  \[ A \oplus B = a \oplus b \oplus c \oplus d \]
- Note that the common terms are cancelled out
Differentials of an s-box

Let $x$ and $x^*$ be the inputs to an s-box

Let $y$ and $y^*$ be the corresponding outputs

Differential Input: $x' = x \oplus x^*$

Differential Output: $y' = y \oplus y^*$

If $x'$ is $(1011)_2$

<table>
<thead>
<tr>
<th>$x$</th>
<th>$x^*$</th>
<th>$y$</th>
<th>$y^*$</th>
<th>$y'$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>1011</td>
<td>1110</td>
<td>1100</td>
<td>0010</td>
</tr>
<tr>
<td>0001</td>
<td>1010</td>
<td>0100</td>
<td>0110</td>
<td>0010</td>
</tr>
<tr>
<td>0010</td>
<td>1001</td>
<td>1101</td>
<td>1010</td>
<td>0111</td>
</tr>
<tr>
<td>0011</td>
<td>1000</td>
<td>0001</td>
<td>0011</td>
<td>0010</td>
</tr>
<tr>
<td>0100</td>
<td>1111</td>
<td>0010</td>
<td>0111</td>
<td>0101</td>
</tr>
<tr>
<td>0101</td>
<td>1110</td>
<td>1111</td>
<td>0000</td>
<td>1111</td>
</tr>
<tr>
<td>0110</td>
<td>1101</td>
<td>1011</td>
<td>1001</td>
<td>0010</td>
</tr>
<tr>
<td>0111</td>
<td>1100</td>
<td>1000</td>
<td>0101</td>
<td>1101</td>
</tr>
<tr>
<td>1000</td>
<td>0011</td>
<td>0011</td>
<td>0001</td>
<td>0010</td>
</tr>
<tr>
<td>1001</td>
<td>0010</td>
<td>1010</td>
<td>1101</td>
<td>0111</td>
</tr>
<tr>
<td>1010</td>
<td>0001</td>
<td>0110</td>
<td>0100</td>
<td>0010</td>
</tr>
<tr>
<td>1011</td>
<td>0000</td>
<td>1100</td>
<td>1110</td>
<td>0010</td>
</tr>
<tr>
<td>1100</td>
<td>0111</td>
<td>0101</td>
<td>1000</td>
<td>1101</td>
</tr>
<tr>
<td>1101</td>
<td>0110</td>
<td>1001</td>
<td>1011</td>
<td>0010</td>
</tr>
<tr>
<td>1110</td>
<td>0101</td>
<td>0000</td>
<td>1111</td>
<td>1111</td>
</tr>
<tr>
<td>1111</td>
<td>0100</td>
<td>0111</td>
<td>0010</td>
<td>0101</td>
</tr>
</tbody>
</table>
Differentials of an s-box

If \( x' \) is \((1011)_2\):

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>1011</td>
<td>1110</td>
<td>1100</td>
<td>0010</td>
</tr>
<tr>
<td>0001</td>
<td>1010</td>
<td>0100</td>
<td>0110</td>
<td>0010</td>
</tr>
<tr>
<td>0010</td>
<td>1001</td>
<td>1101</td>
<td>1010</td>
<td>0111</td>
</tr>
<tr>
<td>0011</td>
<td>1000</td>
<td>0011</td>
<td>0011</td>
<td>0010</td>
</tr>
<tr>
<td>0100</td>
<td>1111</td>
<td>0010</td>
<td>0111</td>
<td>0101</td>
</tr>
<tr>
<td>0101</td>
<td>1110</td>
<td>1111</td>
<td>0000</td>
<td>1111</td>
</tr>
<tr>
<td>0110</td>
<td>1101</td>
<td>1111</td>
<td>1001</td>
<td>0010</td>
</tr>
<tr>
<td>0111</td>
<td>1100</td>
<td>1000</td>
<td>0101</td>
<td>1101</td>
</tr>
<tr>
<td>1000</td>
<td>0011</td>
<td>0011</td>
<td>0011</td>
<td>0010</td>
</tr>
<tr>
<td>1001</td>
<td>0010</td>
<td>1010</td>
<td>1101</td>
<td>0111</td>
</tr>
<tr>
<td>1010</td>
<td>0001</td>
<td>0110</td>
<td>0100</td>
<td>0010</td>
</tr>
<tr>
<td>1011</td>
<td>0000</td>
<td>1100</td>
<td>1110</td>
<td>0010</td>
</tr>
<tr>
<td>1100</td>
<td>0111</td>
<td>0101</td>
<td>1000</td>
<td>1101</td>
</tr>
<tr>
<td>1101</td>
<td>0110</td>
<td>1001</td>
<td>1011</td>
<td>0010</td>
</tr>
<tr>
<td>1110</td>
<td>0101</td>
<td>0000</td>
<td>1111</td>
<td>1111</td>
</tr>
<tr>
<td>1111</td>
<td>0100</td>
<td>0111</td>
<td>0010</td>
<td>0101</td>
</tr>
</tbody>
</table>

Note the non-uniformity..... This non-uniformity is used in differential cryptanalysis
### Differential Distribution Table of the s-box

<table>
<thead>
<tr>
<th>$a'$</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>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>16</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>6</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**$R_p(a', b') = \frac{N_D(a', b')}{2^m}$.**

Probability that output difference is $b'$ given that input difference is $a'$

Counts the number of times input difference is $x'$ and output difference of the s-box is $y'$

$$N_D(x', y') = |\{(x, x^*) \in \Delta(x') : \pi_S(x) \oplus \pi_S(x^*) = y'\}|.$$
Differential trails in a cipher

- First note that the differential output $y'$ does not depend on the secret key.
- Choose a set of consecutive s-boxes so that differences propagate with high propagation ratio. This is the differential trail.

\[
\begin{align*}
\text{In } S_2^1, & \quad R_p(1011, 0010) = 1/2 \\
\text{In } S_2^2, & \quad R_p(0100, 0110) = 3/8 \\
\text{In } S_3^3, & \quad R_p(0010, 0101) = 3/8 \\
\text{In } S_3^3, & \quad R_p(0010, 0101) = 3/8
\end{align*}
\]

- Assuming independence between the s-boxes in the trail, propagation ratio for the trail is the product of individual propagation ratios.

\[
R_p(0000\ 1011\ 0000\ 0000,\ 0000\ 0101\ 0101\ 0000) = \frac{1}{2} \times \left(\frac{3}{8}\right)^3 = \frac{27}{1024}.
\]

- This means that, if the input difference is (0000 1011 0000 0000) then the probability that the output difference is (0000 0101 0101 0000) is 27/1024.
The Differential Cryptanalysis Attack

- The attacker needs
  - A large number of chosen plaintext-ciphertext pairs encrypted with the same key

- The attack
  1. Guess $k^5_{<2>}$ and $k^5_{<4>}$ (256 possibilities)
  2. Compute $v^4_{<2>}$ and $v^4_{<4>}$ for each plaintext–ciphertext using the guessed key
  3. Compute the difference between the inv-sbox($v^4_{<2>}$) and inv-sbox($v^4_{<4>}$)
  4. Test if the required differential is obtained.

If the key guess is correct, the correct differential will be obtained with a probability of $27/1024$

If the key guess is wrong, the differential will be obtained with a probability which is much lower (1/256)
The Differential Cryptanalysis Algorithm

Function inputs are the plaintext-ciphertext Differentials, $T$ is the number of them, and the Inverse of the targeted s-box.

The guessed key $(L_1, L_2)$ is of 256 values.

For each differential, do an initial filtering, and then compute $u^4_{<2>}$ and $u^4_{<4>}$. If these result in the targeted differential $0110, 0110$, then increment the count for the corresponding key guess.

The values of $(L_1, L_2)$ which has the maximum count implies that it is the case where the targeted Differential appears most often. This $(L_1, L_2)$ is the likely key.
DES
(Data Encryption Standard)
History of DES

• Standardized in 1977 by FIPS, as the standard for data encryption
• Based on a Feistel cipher called Lucifer (Lucifer is a Feistel cipher developed by IBM in the early ‘70s)
• NSA made some minor (supposedly controversial) modifications to the Lucifer algorithm
  – Reduced the key size from 64 bits to 56 bits
  – Modifications to the s-boxes
DES Specification

- Block Size: 64 bits
- Key size: 56 bits (+8 parity bits)
- Structure: Fiestel
- Rounds: 16
- Algorithm specifies:
  - encryption / decryption algorithm
  - key expansion algorithm
**DES Initial and Final Permutation**

- Plaintext subjected to an Initial permutation (IP) initially
- After 16 rounds, there is a final permutation (FP) before the ciphertext is generated

Neither operation has any cryptographic significance. Used to facilitate loading of blocks in and out of 1970s eight-bit computer.
IP and FP

Initial Permutation (IP)

The first bit of the o/p is taken from the 58\textsuperscript{th} input bit

Final Permutation (FP = IP\textsuperscript{-1})

This is the inverse of IP
**DES F Function (E and Key mixing)**

- **E** is the expansion block. The 32 bit input is expanded to 48 bits by duplicating some of the bits.
- **key mixing** with subkey.

**Expansion Function**

---

**Diagram Explanation**

- **Plaintext (64 bits)**
- **Expansion Function**
- **Half Block (32 bits)**
- **Subkey (48 bits)**
- **F**
- **for 16 rounds**
- **FP**
- **Cipher Text (64 bits)**
DES F Function (S-boxes)

S1 to S8 are compression s-boxes. Each s-box takes 6 input bits and outputs 4 bits.
DES F Function (Permutation)
DES Key Expansion

- 64 bits input
  - Of which 8 are discarded (or used for parity)
- No non-linear components

<table>
<thead>
<tr>
<th>PC1</th>
<th>Right</th>
</tr>
</thead>
<tbody>
<tr>
<td>57</td>
<td>63</td>
</tr>
<tr>
<td>49</td>
<td>55</td>
</tr>
<tr>
<td>41</td>
<td>47</td>
</tr>
<tr>
<td>33</td>
<td>39</td>
</tr>
<tr>
<td>25</td>
<td>31</td>
</tr>
<tr>
<td>17</td>
<td>23</td>
</tr>
<tr>
<td>9</td>
<td>15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PC1</th>
<th>Left</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>58</td>
<td>7</td>
</tr>
<tr>
<td>50</td>
<td>62</td>
</tr>
<tr>
<td>42</td>
<td>54</td>
</tr>
<tr>
<td>34</td>
<td>46</td>
</tr>
<tr>
<td>26</td>
<td>38</td>
</tr>
<tr>
<td>18</td>
<td>30</td>
</tr>
<tr>
<td>22</td>
<td>22</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PC2</th>
<th>Left</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>28</td>
<td>28</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PC2</th>
<th>Right</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>23</td>
<td>23</td>
</tr>
<tr>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PC2</th>
<th>Left</th>
</tr>
</thead>
<tbody>
<tr>
<td>26</td>
<td>26</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>27</td>
<td>27</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PC2</th>
<th>Right</th>
</tr>
</thead>
<tbody>
<tr>
<td>41</td>
<td>41</td>
</tr>
<tr>
<td>52</td>
<td>52</td>
</tr>
<tr>
<td>31</td>
<td>31</td>
</tr>
<tr>
<td>37</td>
<td>37</td>
</tr>
<tr>
<td>47</td>
<td>47</td>
</tr>
<tr>
<td>55</td>
<td>55</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>32</td>
<td>32</td>
</tr>
</tbody>
</table>

Rotate left

Select 48 out of the 56 bits
DES Decryption

• Same as encryption algorithm, with subkeys applied in reverse order
DES Weak Keys

- In a DES weak key, all the subkeys are the same
  Thus $\text{DES}_{\text{WK}}(\text{DES}_{\text{WK}}(x)) = x$
  (WK is a weak key)
- DES weak keys are as follows

<table>
<thead>
<tr>
<th>56 bit DES weak keys</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000000  0000000</td>
</tr>
<tr>
<td>FFFFFFF  FFFFFFF</td>
</tr>
<tr>
<td>0000000  FFFFFFF</td>
</tr>
<tr>
<td>FFFFFFF  0000000</td>
</tr>
</tbody>
</table>
DES Semi weak keys

- Semi-weak keys have the following properties
  - They appear in pairs: (SK1 and SK1’)
  - $\text{DES}_{SK1}(\text{DES}_{SK1'}(x)) = x$
  - Each semi-weak key has only two sub keys.
### DES Semi weak key pairs

<table>
<thead>
<tr>
<th>First key in the pair</th>
<th>Second key in the pair</th>
</tr>
</thead>
<tbody>
<tr>
<td>01FE 01FE 01FE 01FE</td>
<td>FE01 FE01 FE01 FE01</td>
</tr>
<tr>
<td>1FE0 1FE0 0EF1 0EF1</td>
<td>E01F E01F F10E F10E</td>
</tr>
<tr>
<td>01E0 01E0 01F1 01F1</td>
<td>E001 E001 F101 F101</td>
</tr>
<tr>
<td>1FFE 1FFE 0EFE 0EFE</td>
<td>FE1F FE1F FE0E FE0E</td>
</tr>
<tr>
<td>011F 011F 010E 010E</td>
<td>1F01 1F01 0E01 0E01</td>
</tr>
<tr>
<td>E0FE E0FE F1FE F1FE</td>
<td>FEE0 FEE0 FEF1 FEF1</td>
</tr>
</tbody>
</table>
Objections to DES

• Key size matters
  – Brute Force Attacks due to the small key size

• S-box secrecy
  – During the initial years, the rationale for the DES s-box was kept secret (... to increase security).

• Mathematical attacks :
  – Differential Cryptanalysis
  – Linear Cryptanalysis
DES Cracker

• Specialized ASICs for DES brute force
• Could determine the secret key in less than a day

.... Need to increase key length!!
DES Composition

Key size can be increased by composition

\[ C = \text{DES}_{K_1}(\text{DES}_{K_2}(P)) \]

DES does not form a group under composition.

i.e. It is not possible to obtain

\[ \text{DES}_{K_1}(\text{DES}_{K_2}(P)) = \text{DES}_{K_3}(P) \]

for some key \( K_3 \)

2 DES

keysize = 2*56=112 bits
Meet in the Middle Attack against 2-DES

• Attacker collects a pair of (P,C)
  1. For P, compute $Q_{K_{1}^*} = \text{DES}^{-1}_{K_{1}^*}(P)$ for every possible value of $K_{1}^*$. Record the corresponding $Q_{K_{1}^*}$
  2. For C, compute $Q_{K_{2}^*} = \text{DES}^{-1}_{K_{2}^*}(C)$ for every possible value of $K_{2}^*$. Record the corresponding $Q_{K_{2}^*}$
  3. Find all $K_{1}^*$ and $K_{2}^*$ such that $Q_{K_{1}^*} = Q_{K_{2}^*}$
  4. If Multiple such $K_{1}^*$ and $K_{2}^*$ are found, then repeat with another pair of (P,C)

• Complexity of this attack is $2^{56} + 2^{56} = 2^{57}$
3-DES

- 112 bit security as in 2-DES
- Encrypt → Decrypt → Encrypt
- $K_1 \rightarrow K_2 \rightarrow K_1$ (two 56 bit keys)
- Why EDE and not EEE?
  - Compatibility with the classical DES if $K_1 = K_2$
- Used extensively as a stopgap arrangement until a new cipher standard (AES) was established
- Drawbacks of 3-DES:
  - Sluggish in software
  - Could only encrypt 64 bit blocks at a time
How to choose a good s-box?
Criteria for a good s-box

• Completeness
• Balance
• Non-linearity
• Propagation criteria
• Good XOR profile
• High Algebraic Degree
Sboxes

• In an s-box each output bit can be represented as a **Boolean function** of its input bits:

\[
\begin{align*}
y_1 &= f_1(x_1, x_2, x_3, \cdots, x_m) \\
y_2 &= f_2(x_1, x_2, x_3, \cdots, x_m) \\
y_3 &= f_3(x_1, x_2, x_3, \cdots, x_m) \\
&\vdots \\
y_n &= f_n(x_1, x_2, x_3, \cdots, x_m)
\end{align*}
\]

The functions have to be non-linear. Linear functions are easily reversed.
Boolean Functions

- A Boolean function is a mapping from \( \{0,1\}^m \rightarrow \{0,1\} \)
- **Algebraic Normal Form** representation of a Boolean function
  - A Boolean function on \( m \)-inputs can be represented with sum (XOR +) of products (AND .) form:
    \[
    y = a_0 \oplus a_1x_1 \oplus a_2x_2 \oplus a_3x_1x_2
    \]
    where \( a_i \) is either 0 or 1.

- **Affine Form**: if all the AND terms have coefficients 0
- **Linear form**: Affine form and \( a_0 = 0 \)
Truth Tables

- Consider a Boolean function $f : \{0,1\}^m \rightarrow \{0,1\}$
- The following Binary sequence is the truth table of $f$

\[
(f(\alpha_0), f(\alpha_1), f(\alpha_2), \ldots, f(\alpha_{2^m-1}))
\]

where are $m$ bit numbers and $\alpha_i \neq \alpha_j$ unless $i = j$

- The truth table is therefore $(0,1,1,1)$

\[
f : y = x_1 \oplus x_2 \oplus x_1 x_2
\]
Balanced Boolean Functions

- A Boolean function is said to be balanced if its truth table has an equal number of 0s and 1s.
- S-box equations should be balanced (i.e., 0 and 1 have an equal probability of occurrence).

\[
\begin{align*}
\text{Unbalanced function} & : y = x_1 \oplus x_2 \oplus x_1x_2 \\
\text{Balanced function} & : y = x_1 \oplus x_2
\end{align*}
\]

<table>
<thead>
<tr>
<th>X1</th>
<th>X2</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>X1</th>
<th>X2</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Distance Between functions

Let $f$ and $g$ be two Boolean functions
Let $\eta$ be the truth table for $f$ and $\varepsilon$ the truth table for $g$

$$HD(\eta, \varepsilon)$$ is the Hamming distance between the two sequences

$$f : y_1 = x_1 \oplus x_2 \oplus x_1 x_2$$

$$g : y_2 = x_1 \oplus x_2$$

$$HD(\eta, \varepsilon) = 2$$
Nonlinearity of a Boolean Function

- The non-linearity of a Boolean function is the **minimum distance between the function and the set of all affine functions**.
  - Strengthens against linear cryptanalysis

\[
y_1 = x_1 \oplus x_2 \oplus x_1x_2
\]
\[
y_2 = 0
\]
\[
y_3 = x_1
\]
\[
y_4 = x_2
\]
\[
y_5 = x_1 \oplus x_2
\]

<table>
<thead>
<tr>
<th></th>
<th>X1</th>
<th>X2</th>
<th>Y1</th>
<th>Y2</th>
<th>Y3</th>
<th>Y4</th>
<th>Y5</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>X2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>X3</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>X4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**Nonlinearity:** \( N_f = \text{MIN}_{g \in \text{Affine}}(HD(f, g)) \)

**Nonlinearity of \( y_1 \):** \( N_{y_1} = 1 \)
On the Non-linearity of Boolean Functions

- HD of any two linear functions is $2^{n-1}$
- HD between linear functions and a non-linear function is $< 2^{n-1}$

Let $\xi = \#(f = g) - \#(f \neq g)$

$$= 2^n - \#(f \neq g) - \#(f \neq g)$$

$$= 2^n - 2\#(f \neq g)$$

$$HD(f, g) = \#(f \neq g) = 2^{n-1} - \frac{1}{2} \xi$$
Bent Functions

• Bent functions are non-linear Boolean functions which have maximum non-linearity
• The non-linearity of a Bent function is $2^{n-1} - 2^{\frac{n}{2}-1}$
• They satisfy SAC but are **not balanced**

• Example : $f(x) = x_1x_2 + x_3x_4$
Walsh Hadamard Matrix

• A compact combinatorial representation of all affine functions
• Each row of the WH matrix forms the truth table of all affine functions with N variables can be represented by the matrix

\[
H(2^N) = \begin{bmatrix}
H(2^{N-1}) & H(2^{N-1}) \\
H(2^{N-1}) & \text{complement}(H(2^{N-1}))
\end{bmatrix}
\]

\[
H(2^1) = \begin{bmatrix}
0 & 0 \\
0 & 1
\end{bmatrix}
\]

\[
H(2^2) = \begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 \\
0 & 1 & 1 & 0
\end{bmatrix}
\]
Affine Transformations and Non-linearity

- If a Boolean function is **balanced**, then an affine transformation does not affect its non-linearity.

\[ f(x) \] is a balanced Boolean function, then \[ f(xB \oplus A) \] is also balanced

\[ x = (x_1, x_2, x_3, ..., x_n) \]

- \( B \) is a \( n \times n \) binary invertible matrix
- \( A \) is an \( n \) bit vector

The nonlinearity of \( f(x) = \) nonlinearity of \( f(xB \oplus A) \)
Strict Avalanche Criteria (SAC)

- For a function \( f \) to satisfy SAC,
  \[ f(x) \oplus f(x \oplus \alpha) \] must be balanced, for any \( \alpha \) with \( HW(\alpha) = 1 \)

- Also called *propagation criteria of order 1*

- Higher order SAC,
  - Propagation criteria of order \( > 1 \)
  - When input changes in more than 1 bit

- Show that
  \[ y = x_1 x_2 \oplus x_3 \] does not satisfy SAC
  \[ z = x_1 x_2 \oplus x_3 x_4 \] satisfies SAC

  Note that \( z \) is a Bent function
How to make a Boolean function satisfy SAC

• Let $f(x)$ be a Boolean function of order $n$
• Let $A$ be an $n \times n$ non-singular Boolean matrix
• If $r$ is a row in the matrix $A$ and $f(x) \oplus f(x \oplus r)$ is balanced then $g(x) = f(xA)$ satisfies SAC

Example:

$$f = x_1 x_2 \oplus x_3$$

$$A = \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
1 & 1 & 1
\end{bmatrix}$$

then $g(x) = f(xA)$ satisfies SAC

verify this?
Completeness

• More a criteria for the complete cipher (SP)
• Given s-boxes with a fixed mapping,
  – P-layer needs to be fixed and rounds need to be fixed such that ciphertext is a complex function of every plaintext input
XOR Profile

• The difference distribution table of the s-box must contain small variations
Modes of Operation
What are Modes of Operation?

• Block cipher algorithms only encrypt a single block of message
• A mode of operation describes how to repeatedly apply a cipher's single-block operation to securely transform amounts of data larger than a block
• Modes of Operation
  – Electronic code book mode (ECB Mode)
  – Cipher feedback mode (CFB Mode)
  – Cipher block chaining mode (CBC mode)
  – Output feedback mode (OFB mode)
  – Counter mode
ECB Mode

- Every block in the message is encrypted independently with the same key
- Drawback 1: If $p_i = p_j$ ($i \neq j$) then $c_i = c_j$
  - Encryption should protect against known plaintext attacks (since the attacker could guess parts of the message..... Like stereotype beginnings)
- Drawback 2: An interceptor may alter the order of the blocks during transmission
- Not recommended for encryption of more than one block
CBC Mode

- Cipher Block Chaining
- **Advantage 1**: Encryption dependent on a previous the ciphertext of a previous block, therefore
  - $c_i \neq c_j \ (i \neq j)$ even if $p_i = p_j$
- **Advantage 2**: Intruder cannot alter the order of the blocks during transmission
- If an error is present in one received block (say $c_i$)
  - Then $c_i$ and $c_{i+1}$ will not be decrypted correctly
  - All remaining blocks will be correctly decrypted
CBC Mode Decryption

\[ p_0 \xrightarrow{\text{IV}} c_0 \xrightarrow{e_K} c_1 \xrightarrow{e_K} c_2 \xrightarrow{e_K} c_3 \xrightarrow{e_K} c_4 \]

\[ c_0 \xrightarrow{d_K} p_0 \xrightarrow{\text{IV}} c_1 \xrightarrow{d_K} c_2 \xrightarrow{d_K} c_3 \xrightarrow{d_K} c_4 \]
CFB (Cipher feedback Mode)

Can transform a block cipher into a stream cipher.
   – i.e. Each block encrypted with a different key

Uses a shift register that is initialized with an IV
CFB - Error Propagation

Uses a shift register that is initialized with an IV
Previous ciphertext block fed into shift register

Decryption Scheme
Output Feedback Mode (OFB)

- Very similar to CFB but feedback taken from output of $e_k$
- An error in one byte of the ciphertexts affects only one decryption

Encryption Scheme
(Decryption scheme is similar)
Counter Mode

- A randomly initialized counter is incremented with every encryption
- Can be parallelized
  - I.e. Multiple encryption engines can simultaneously run
- As with OFB, an error in a single ciphertext block affects only one decrypted plaintext
The Advanced Encryption Standard (AES)
**Advanced Encryption Standard (AES)**

- NIST’s standard for block cipher since October 2000.

<table>
<thead>
<tr>
<th></th>
<th>Key Length</th>
<th>No. of rounds</th>
</tr>
</thead>
<tbody>
<tr>
<td>AES-128</td>
<td>16 bytes</td>
<td>10</td>
</tr>
<tr>
<td>AES-192</td>
<td>24 bytes</td>
<td>12</td>
</tr>
<tr>
<td>AES-256</td>
<td>32 bytes</td>
<td>14</td>
</tr>
</tbody>
</table>

- SPN network with each round having
  - Randomness Layer: *Round key addition*
  - Confusion Layer: *Byte Substitution*
  - Diffusion Layer: *Shift row and Mix column*
  (the last round does not have mix column step)
Mathematical Background

Finite Fields
The AES State Representation

16 byte plaintext

![Matrix](image)

16 byte ciphertext

- 16 bytes arranged in a 4x4 matrix of bytes
AES-128 Encryption

4 Operations
- Byte Substitution
- Shift Rows
- Mix Columns
- Add Round Key

Loop 10 times

XOR key

Byte Substitution

Shift Rows

Mix Columns (except for the last round)

Add Round Key

Key Expansion

Secret Key

Plaintext Block

Ciphertext Block
AES-128 Encryption

- Plaintext Block
- XOR key
- Byte Substitution
- Shift Rows
- Mix Columns (except for the last round)
- Add Round Key
- Key Expansion

102 Loops

confusion

diffusion
AES Operations

• All AES operations are performed in the field GF(2^8).

• The field’s irreducible polynomial is

\[ x^8 + x^4 + x^3 + x + 1 \]

in binary notation \((1\,0001\,1011)_2\)

in hex notation \((11B)_{16}\)
Byte Substitution

- Makes a non-linear substitution for every byte in the 4x4 matrix

\[
Sbox(A) = \begin{cases} 
\text{Affine}(A^{-1}) & \text{if } A(\theta) \neq 0 \\
\text{Affine}(0) & \text{if } A(\theta) = 0
\end{cases}
\]
AES S-box Design Rationale

\[
Sbox(A) = \begin{cases} 
\text{Affine}(A^{-1}) & \text{if } A(\theta) \neq 0 \\
\text{Affine}(0) & \text{if } A(\theta) = 0 
\end{cases}
\]

• This s-box construction was proposed by Kaiser Nyberg in 1993

• Steps:
  1. **Inverse in GF(2^8)**
     • Provides high degrees of non-linearity
     • Known to have good resistance against differential and linear cryptanalysis
  2. **Affine transformation**
     • ensures no fixed points : i.e. Fixed points : \( S(x) = x \)
     • Complicates Algebraic attacks
S-box Encryption Table

- Use a table to do the byte substitution
- eg. $\text{Sbox}[42] = 2c$

<table>
<thead>
<tr>
<th>x</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>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0c</td>
<td>7c</td>
<td>77</td>
<td>7b</td>
<td>e2</td>
<td>6b</td>
<td>c6</td>
<td>c5</td>
<td>30</td>
<td>91</td>
<td>07</td>
<td>2b</td>
<td>fe</td>
<td>d7</td>
<td>ab</td>
<td>7c</td>
</tr>
<tr>
<td>1</td>
<td>ca</td>
<td>82</td>
<td>c9</td>
<td>7d</td>
<td>fa</td>
<td>59</td>
<td>47</td>
<td>f0</td>
<td>ad</td>
<td>d4</td>
<td>a2</td>
<td>af</td>
<td>9c</td>
<td>a4</td>
<td>72</td>
<td>c0</td>
</tr>
<tr>
<td>2</td>
<td>b7</td>
<td>fd</td>
<td>93</td>
<td>25</td>
<td>36</td>
<td>3f</td>
<td>f7</td>
<td>cc</td>
<td>34</td>
<td>a5</td>
<td>e5</td>
<td>f1</td>
<td>71</td>
<td>d8</td>
<td>31</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>04</td>
<td>c7</td>
<td>23</td>
<td>c3</td>
<td>18</td>
<td>96</td>
<td>05</td>
<td>9a</td>
<td>07</td>
<td>12</td>
<td>80</td>
<td>e2</td>
<td>eb</td>
<td>27</td>
<td>b2</td>
<td>75</td>
</tr>
<tr>
<td>4</td>
<td>0b</td>
<td>92</td>
<td>3c</td>
<td>1a</td>
<td>56</td>
<td>5a</td>
<td>50</td>
<td>52</td>
<td>9b</td>
<td>d6</td>
<td>b3</td>
<td>29</td>
<td>e1</td>
<td>2f</td>
<td>81</td>
<td></td>
</tr>
</tbody>
</table>

- $x$ is the input byte
- $y$ is the output byte via the S-box

Values in the table correspond to hexadecimal representations of bytes.
• **ShiftRows**
  - Leave the First row untouched
  - Left Rotate (2\textsuperscript{nd} Row by 8 bits)
  - Left Rotate (3\textsuperscript{rd} Row by 16 bits)
  - Left Rotate (4\textsuperscript{th} Row by 24 bits)
• Along with MixColumns provides high diffusion
  - Bits flip in at-least 25 s-boxes after 4 rounds
Mix Columns

The 4x4 matrix is multiplied with the matrix

\[
\begin{bmatrix}
2 & 3 & 1 & 1 \\
1 & 2 & 3 & 1 \\
1 & 1 & 2 & 3 \\
3 & 1 & 1 & 2
\end{bmatrix}
\times
\begin{bmatrix}
a & e & i & m \\
b & f & j & n \\
c & g & k & o \\
d & h & l & p
\end{bmatrix}
\]

Note that multiplications are in GF(2^8) field.

\[
E = 2e + 3f + g + h \\
F = e + 2f + 3g + h \\
G = e + f + 2g + 3h \\
H = 3e + f + g + 2h
\]
Mix Columns Rationale

Why use this matrix?

• It is an MDS matrix (Maximum Distance Separable codes)
  – If the input of a column changes then all outputs change
  – This maximizes the branch number
  – For AES, the branch number is 5

• Values [2,3,1,1], are the smallest which result in MDS matrix that is also circulant

• Has an inverse in the AES field
**AES Operations**

(Add Round Key)

Addition here is addition in GF($2^8$), which is the ex-or operation.
AES-128 Decryption

Ciphertext Block

Inverse Mix Columns (except for the last round)

Add Round Key

Inverse Shift Rows

Inverse Byte Substitution

XOR RK10

Key Expansion

RK9
RK8
RK1
key

Secret Key

Plaintext Block

Loop 10 times
Inverse S-box

- Simply the AES s-box run in reverse
- As with the s-box operation, a lookup table can be used

<table>
<thead>
<tr>
<th>0x</th>
<th>x0</th>
<th>x1</th>
<th>x2</th>
<th>x3</th>
<th>x4</th>
<th>x5</th>
<th>x6</th>
<th>x7</th>
<th>x8</th>
<th>x9</th>
<th>xa</th>
<th>xb</th>
<th>xc</th>
<th>xd</th>
<th>xe</th>
<th>xf</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>52</td>
<td>09</td>
<td>6a</td>
<td>d5</td>
<td>30</td>
<td>3C</td>
<td>a5</td>
<td>30</td>
<td>bf</td>
<td>40</td>
<td>a3</td>
<td>9e</td>
<td>01</td>
<td>f3</td>
<td>d7</td>
<td>fb</td>
</tr>
<tr>
<td>1</td>
<td>7c</td>
<td>e3</td>
<td>39</td>
<td>82</td>
<td>9b</td>
<td>2f</td>
<td>ff</td>
<td>87</td>
<td>34</td>
<td>8e</td>
<td>43</td>
<td>44</td>
<td>c4</td>
<td>de</td>
<td>e9</td>
<td>cb</td>
</tr>
<tr>
<td>2</td>
<td>54</td>
<td>7b</td>
<td>94</td>
<td>32</td>
<td>a6</td>
<td>c2</td>
<td>23</td>
<td>3d</td>
<td>ee</td>
<td>4c</td>
<td>95</td>
<td>0b</td>
<td>42</td>
<td>fa</td>
<td>c3</td>
<td>4e</td>
</tr>
<tr>
<td>3</td>
<td>08</td>
<td>2e</td>
<td>a1</td>
<td>66</td>
<td>28</td>
<td>d9</td>
<td>24</td>
<td>b2</td>
<td>76</td>
<td>5b</td>
<td>a2</td>
<td>49</td>
<td>6d</td>
<td>8b</td>
<td>d1</td>
<td>25</td>
</tr>
<tr>
<td>4</td>
<td>72</td>
<td>f8</td>
<td>f6</td>
<td>64</td>
<td>86</td>
<td>68</td>
<td>98</td>
<td>16</td>
<td>d4</td>
<td>a4</td>
<td>5c</td>
<td>cc</td>
<td>5d</td>
<td>65</td>
<td>b6</td>
<td>92</td>
</tr>
<tr>
<td>5</td>
<td>6c</td>
<td>70</td>
<td>48</td>
<td>50</td>
<td>fd</td>
<td>ed</td>
<td>b9</td>
<td>da</td>
<td>5e</td>
<td>15</td>
<td>46</td>
<td>57</td>
<td>a7</td>
<td>8d</td>
<td>9d</td>
<td>84</td>
</tr>
<tr>
<td>6</td>
<td>90</td>
<td>d8</td>
<td>ab</td>
<td>00</td>
<td>8c</td>
<td>bc</td>
<td>d3</td>
<td>0a</td>
<td>f7</td>
<td>e4</td>
<td>58</td>
<td>05</td>
<td>b8</td>
<td>b3</td>
<td>45</td>
<td>06</td>
</tr>
<tr>
<td>7</td>
<td>d0</td>
<td>2c</td>
<td>1e</td>
<td>8f</td>
<td>ca</td>
<td>3f</td>
<td>0f</td>
<td>02</td>
<td>c1</td>
<td>af</td>
<td>bd</td>
<td>03</td>
<td>01</td>
<td>13</td>
<td>8a</td>
<td>6b</td>
</tr>
<tr>
<td>8</td>
<td>3a</td>
<td>91</td>
<td>11</td>
<td>41</td>
<td>4f</td>
<td>67</td>
<td>dc</td>
<td>ea</td>
<td>97</td>
<td>f2</td>
<td>cf</td>
<td>ce</td>
<td>f0</td>
<td>b4</td>
<td>e6</td>
<td>73</td>
</tr>
<tr>
<td>9</td>
<td>96</td>
<td>ac</td>
<td>74</td>
<td>22</td>
<td>e7</td>
<td>ad</td>
<td>35</td>
<td>05</td>
<td>e2</td>
<td>f9</td>
<td>37</td>
<td>e0</td>
<td>1c</td>
<td>75</td>
<td>df</td>
<td>6c</td>
</tr>
<tr>
<td>a</td>
<td>47</td>
<td>f1</td>
<td>1a</td>
<td>71</td>
<td>1d</td>
<td>29</td>
<td>c5</td>
<td>89</td>
<td>6f</td>
<td>b7</td>
<td>62</td>
<td>0e</td>
<td>aa</td>
<td>18</td>
<td>be</td>
<td>1b</td>
</tr>
<tr>
<td>b</td>
<td>fc</td>
<td>56</td>
<td>3e</td>
<td>4b</td>
<td>c6</td>
<td>d2</td>
<td>79</td>
<td>20</td>
<td>9a</td>
<td>db</td>
<td>c0</td>
<td>fe</td>
<td>78</td>
<td>cd</td>
<td>5a</td>
<td>f4</td>
</tr>
<tr>
<td>c</td>
<td>1f</td>
<td>dd</td>
<td>a8</td>
<td>33</td>
<td>88</td>
<td>07</td>
<td>c7</td>
<td>31</td>
<td>b1</td>
<td>12</td>
<td>10</td>
<td>59</td>
<td>27</td>
<td>80</td>
<td>ec</td>
<td>5f</td>
</tr>
<tr>
<td>d</td>
<td>60</td>
<td>51</td>
<td>7f</td>
<td>a9</td>
<td>19</td>
<td>b5</td>
<td>4a</td>
<td>0d</td>
<td>2d</td>
<td>e5</td>
<td>7a</td>
<td>9f</td>
<td>93</td>
<td>c9</td>
<td>9c</td>
<td>ef</td>
</tr>
<tr>
<td>e</td>
<td>a0</td>
<td>e0</td>
<td>3b</td>
<td>4d</td>
<td>ae</td>
<td>2a</td>
<td>f5</td>
<td>b0</td>
<td>c8</td>
<td>eb</td>
<td>bb</td>
<td>3c</td>
<td>83</td>
<td>53</td>
<td>99</td>
<td>61</td>
</tr>
<tr>
<td>f</td>
<td>17</td>
<td>2b</td>
<td>04</td>
<td>7e</td>
<td>ba</td>
<td>77</td>
<td>d6</td>
<td>26</td>
<td>e1</td>
<td>69</td>
<td>14</td>
<td>63</td>
<td>55</td>
<td>21</td>
<td>0c</td>
<td>7d</td>
</tr>
</tbody>
</table>
Inverse Shift Rows

- **ShiftRows**
  - Leave the First row untouched
  - Right Rotate (2\textsuperscript{nd} Row by 8 bits)
  - Right Rotate (3\textsuperscript{rd} Row by 16 bits)
  - Right Rotate (4\textsuperscript{th} Row by 24 bits)
Inverse Mix Column

\[
\begin{align*}
E &= Ee + Bf + Dg + 9h \\
F &= 9e + Ef + Bg + Dh \\
G &= De + 9f + Eg + Dh \\
H &= Be + Df + 9g + Eh
\end{align*}
\]

- The 4x4 matrix is multiplied with the matrix
  \[
  \begin{bmatrix}
    E & B & D & 9 \\
    9 & E & B & D \\
    D & 9 & E & B \\
    B & D & 9 & E
  \end{bmatrix}
  \]
- The hardware implementation can be done in a similar way as mix columns
AES Key Schedule

- How to expand the secret key
- Design Criteria
  - Efficient
  - Non-symmetric: Ensured by round constants
  - Efficient diffusion properties of secret key into round keys
  - It should exhibit enough non-linearity to prohibit the full determination of differences in the expanded key from cipher key differences only.
AES Key Schedule

secret key

1\textsuperscript{st} round key

1\textsuperscript{st} round key

2^{i-1}

0

0

0
Implementation Aspects of AES
Software Implementations of AES Encryption

- S-box implemented as a lookup-table (256 bytes)
- Shift rows combined with Mix columns
- Multiplication with MDS matrix easily achieved
  - $x^2$, done by left shift. If there is an overflow an ex-or with 0x1B is needed
  - $x^3 = x^2 + x$
AES on 32 bit Systems

AES state

\[
\begin{bmatrix}
a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\
a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\
a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\
a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3}
\end{bmatrix}
\]

Byte Substitution

\[b_{i,j} = S(a_{i,j}) \text{ for } i, j \in \{0,1,2,3\}\]

Shift Rows

(c1 = c2 = c3 = 1 are cyclic shifts)

\[
\begin{bmatrix}
c_{0,j} \\
c_{1,j} \\
c_{2,j} \\
c_{3,j}
\end{bmatrix}
= \begin{bmatrix}
b_{0,j} \\
b_{1,\text{C1}-j} \\
b_{2,\text{C2}-j} \\
b_{3,\text{C3}-j}
\end{bmatrix}
\]

Mix Columns

\[
\begin{bmatrix}
d_{0,j} \\
d_{1,j} \\
d_{2,j} \\
d_{3,j}
\end{bmatrix}
= \begin{bmatrix}
02 & 03 & 01 & 01 \\
01 & 02 & 03 & 01 \\
01 & 01 & 02 & 03 \\
03 & 01 & 01 & 02
\end{bmatrix}
\begin{bmatrix}
c_{0,j} \\
c_{1,j} \\
c_{2,j} \\
c_{3,j}
\end{bmatrix}
\]

Add Round Key

\[e_{i,j} = d_{i,j} \oplus k_{i,j} \text{ for } i, j \in \{0,1,2,3\}\]

Combining Operations

\[
\begin{bmatrix}
e_{0,j} \\
e_{1,j} \\
e_{2,j} \\
e_{3,j}
\end{bmatrix}
= \begin{bmatrix}
02 & 03 & 01 & 01 \\
01 & 02 & 03 & 01 \\
01 & 01 & 02 & 03 \\
03 & 01 & 01 & 02
\end{bmatrix}
\begin{bmatrix}
S[a_{0,j}] \\
S[a_{1,j-C1}] \\
S[a_{2,j-C2}] \\
S[a_{3,j-C3}]
\end{bmatrix}
\oplus
\begin{bmatrix}
k_{0,j} \\
k_{1,j} \\
k_{2,j} \\
k_{3,j}
\end{bmatrix}
\]
T Tables

Combining Operations

\[
\begin{bmatrix}
    e_{0,j} \\
    e_{1,j} \\
    e_{2,j} \\
    e_{3,j}
\end{bmatrix} = S[a_{0,j}] \oplus S[a_{1,j} \cdot c_1] \oplus S[a_{2,j} \cdot c_2] \oplus S[a_{3,j} \cdot c_3] \oplus k_{0,j} \oplus k_{1,j} \oplus k_{2,j} \oplus k_{3,j}
\]

Define 4 T-Tables

\[
T_0[a] = \begin{bmatrix}
    S[a] \cdot 02 \\
    S[a] \\
    S[a] \\
    S[a] \cdot 03
\end{bmatrix},
T_1[a] = \begin{bmatrix}
    S[a] \cdot 03 \\
    S[a] \cdot 02 \\
    S[a] \\
    S[a] \cdot 03
\end{bmatrix},
T_2[a] = \begin{bmatrix}
    S[a] \\
    S[a] \cdot 03 \\
    S[a] \cdot 02 \\
    S[a]
\end{bmatrix},
T_3[a] = \begin{bmatrix}
    S[a] \\
    S[a] \\
    S[a] \cdot 03 \\
    S[a] \cdot 02
\end{bmatrix}
\]

One Round of AES using T-Tables

\[
e_j = T_0[a_{0,j}] \oplus T_1[a_{1,j} \cdot c_1] \oplus T_2[a_{2,j} \cdot c_2] \oplus T_3[a_{3,j} \cdot c_3] \oplus k_j.
\]
OpenSSL Implementation of AES (with T-tables)

```c
static const u32 Te0[256] = {
    0xc66363a5U, 0xf87c7c84U, 0xee777799U, 0xf67b7b8dU,
    0xffff2f20dU, 0xd66b66bdU, 0xe6f6f1bU, 0x91c5c554U,
    0x60303050U, 0x02010103U, 0xe6767a9U, 0x562b2b7dU,
};

static const u32 Te1[256] = {
    0xa5c66363U, 0x84f87c7cU, 0x99ee7777U, 0xdf67b7bU,
    0x0dfff2f2U, 0xbdd66bd6bU, 0xb1de6f6fU, 0x5491c554U,
};

static const u32 Te2[256] = {
    0x63a5c663U, 0x7c84f87cU, 0x7799ee77U, 0x7b86f67bU,
    0xf20dfff2U, 0xbdd66bd6bU, 0x6f6b1de6fU, 0xc55491c5U,
};

static const u32 Te3[256] = {
    0x6363a5c6U, 0x7c7c84f8U, 0x777799eeU, 0x7b7b86f6U,
    0xf20dfff2U, 0xb6b6b6bd6U, 0x6f6b1de6fU, 0xc5c55491U,
    0x36305060U, 0x01010302U, 0x6767a9ceU, 0x2b2b7d56U,
};
```

```c
s0 = GETU32(in) ^ rk[0];
s1 = GETU32(in + 4) ^ rk[1];
s2 = GETU32(in + 8) ^ rk[2];
s3 = GETU32(in + 12) ^ rk[3];

/* round 1: */
t0 = Te0[s0 >> 24] ^ Te1[(s1 >> 16) & 0xff] ^ Te2[(s2 >> 8) & 0xff] ^ Te3[(s3 & 0xff) & rk[4];
t1 = Te0[s1 >> 24] ^ Te1[(s2 >> 16) & 0xff] ^ Te2[(s3 >> 8) & 0xff] ^ Te3[(s0 & 0xff) & rk[5];
t2 = Te0[s2 >> 24] ^ Te1[(s3 >> 16) & 0xff] ^ Te2[(s0 >> 8) & 0xff] ^ Te3[(s1 & 0xff) & rk[6];
t3 = Te0[s3 >> 24] ^ Te1[(s0 >> 16) & 0xff] ^ Te2[(s1 >> 8) & 0xff] ^ Te3[(s2 & 0xff) & rk[7];

/* round 2: */
s0 = Te0[t0 >> 24] ^ Te1[(t1 >> 16) & 0xff] ^ Te2[(t2 >> 8) & 0xff] ^ Te3[(t3 & 0xff) & rk[8];
s1 = Te0[t1 >> 24] ^ Te1[(t2 >> 16) & 0xff] ^ Te2[(t3 >> 8) & 0xff] ^ Te3[(t0 & 0xff) & rk[9];
s2 = Te0[t2 >> 24] ^ Te1[(t3 >> 16) & 0xff] ^ Te2[(t0 >> 8) & 0xff] ^ Te3[(t1 & 0xff) & rk[10];
s3 = Te0[t3 >> 24] ^ Te1[(t0 >> 16) & 0xff] ^ Te2[(t1 >> 8) & 0xff] ^ Te3[(t2 & 0xff) & rk[11];
```
Last Round of AES

- Uses a different table (Te4)

```c
s0 =
    (Te4[(t0 >> 24) ] & 0xff000000) ^
    (Te4[(t1 >> 16) & 0xff] & 0x00ff0000) ^
    (Te4[(t2 >> 8 ) & 0xff] & 0x0000ff00) |
    (Te4[(t3 ) & 0xff] & 0x000000ff) ^
    rk[0];
PUTU32(out , s0);

s1 =
    (Te4[(t1 >> 24) ] & 0xff000000) ^
    (Te4[(t2 >> 16) & 0xff] & 0x00ff0000) ^
    (Te4[(t3 >> 8 ) & 0xff] & 0x0000ff00) ^
    (Te4[(t0 ) & 0xff] & 0x000000ff) ^
    rk[1];
PUTU32(out +  4, s1);

s2 =
    (Te4[(t2 >> 24) ] & 0xff000000) ^
    (Te4[(t3 >> 16) & 0xff] & 0x00ff0000) ^
    (Te4[(t0 >> 8 ) & 0xff] & 0x0000ff00) ^
    (Te4[(t1 ) & 0xff] & 0x000000ff) ^
    rk[2];
PUTU32(out +  8, s2);

s3 =
    (Te4[(t3 >> 24) ] & 0xff000000) ^
    (Te4[(t0 >> 16) & 0xff] & 0x00ff0000) ^
    (Te4[(t1 >> 8 ) & 0xff] & 0x0000ff00) ^
    (Te4[(t2 ) & 0xff] & 0x000000ff) ^
    rk[3];
PUTU32(out + 12, s3);
```
AES NI

- Accelerating AES on modern Intel and AMD processors with dedicated instructions

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AESENC</td>
<td>Perform one round of an AES encryption flow</td>
</tr>
<tr>
<td>AESENCLAST</td>
<td>Perform the last round of an AES encryption flow</td>
</tr>
<tr>
<td>AESDEC</td>
<td>Perform one round of an AES decryption flow</td>
</tr>
<tr>
<td>AESDECLAST</td>
<td>Perform the last round of an AES decryption flow</td>
</tr>
<tr>
<td>AESKEYGENASSIST</td>
<td>Assist in AES round key generation</td>
</tr>
<tr>
<td>AESIMC</td>
<td>Assist in AES Inverse Mix Columns</td>
</tr>
<tr>
<td>PCLMULQDQ</td>
<td>Carryless multiply (CLMUL)</td>
</tr>
</tbody>
</table>
Compact Implementations of AES

• How should the S-box be implemented?
  – Look up table (256 bytes)
    • This may be too large for some devices
  – Finding the inverse (using Itoh-Tsujii or the extended Euclidean algorithm) and then affine transformation
    • Again expensive (too big!!!)
  – Third alternative
    • Use composite fields
Composite Fields
(refer Math. Background)
Composite Fields for AES

• The AES Field is \( GF(2^8)/x^8+x^4+x^3+x+1 \)
  – Has order 256

• Many composite fields for AES exists
  – \( GF(2^4)^2 \)
    • Requires two irreducible polynomials
      One has the form \( x^4 + \ldots \), where coefficients are in \( GF(2) \)
      The second has the form \( x^2 + ax + b \), where \( a, b \) are in \( GF(2^4) \)
  – \( GF((2^2)^2)^2 \)
    • Requires three irreducible polynomials
      First of the form \( x^2 + a_1x + b_1 \), where \( a_1, b_1 \) in \( GF(2) \)
      Second has the form \( x^2 + a_2x + b_2 \), where \( a_2, b_2 \) in \( GF(2^2) \)
      Third has the form \( x^2 + a_3x + b_3 \), where \( a_3, b_3 \) in \( GF(2^2)^2 \)
Mapping between GF(2^8) and Composite Fields

FindMap()
{
    Initilize MAP[0] = 0 and REVMAP[0] = 0
    Find $\alpha$ a primitive root of field GF(2^8)
    Find $\beta$ a primitive root of field GF(2^4)^2
    $\alpha' = 1$; $\beta' = 1$
    For $i = 1$ to 255
    
    $\alpha' = \alpha \cdot \alpha'$  (Multiplication in the field GF(2^8))
    $\beta' = \beta \cdot \beta'$  (Multiplication in the field GF(2^4)^2)
    $MAP[\alpha'] = \beta'$
    $REVMAP[\beta'] = \alpha'$

    return MAP and REVMAP
}

https://drive.google.com/file/d/0BwxUBZXYoUKCTEJmNUozMI9xM3M/edit?usp=sharing
Implementing the AES S-box in Composite Fields

\[ \text{x} \xrightarrow{\text{Map}} \text{Inverse in Composite Field} \quad \text{eg. In } \mathbb{GF}(2^4)^2 \xrightarrow{\text{Reverse Map}} \text{Affine Transform} \xrightarrow{\text{Sbox}(x)} \]
S-box Based on Composite Fields

### Gate Count for composite Sbox#

<table>
<thead>
<tr>
<th>XOR</th>
<th>NAND</th>
<th>NOR</th>
<th>Total Gates in terms of NAND (using std cell lib)</th>
</tr>
</thead>
<tbody>
<tr>
<td>80</td>
<td>34</td>
<td>6</td>
<td>180</td>
</tr>
</tbody>
</table>

### Performance of S-boxes on FPGA*:

<table>
<thead>
<tr>
<th>S-box Approach</th>
<th>No. of Slices</th>
<th>Critical Path</th>
<th>Gate Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lookup table based</td>
<td>64</td>
<td>11.9ns</td>
<td>1128</td>
</tr>
<tr>
<td>Composite Field based</td>
<td>30</td>
<td>18.3ns</td>
<td>312</td>
</tr>
</tbody>
</table>

# D. Canright, A Very Compact S-box for AES, CHES-2005
* Simulation Results using Xilinx ISE
Overhead of Composite Field s-boxes

• Composite field s-boxes require mapping and reverse mapping to and from the composite fields in each round

• An alternate approach is to convert all other round operations into composite field operations.
  – This would require just one mapping and one reverse mapping for the entire encryption
  – Operations Add Round Key and Shift Rows are not altered.
  – Mix Columns will need to be re-implemented
Attacks on AES
Differential and Linear Properties of AES

• Differential Cryptanalysis
  – No 4 round differential trail > $1/2^{150}$ and no 8 round differential trail > $1/2^{300}$ exists.

• Linear Cryptanalysis
  – No 4 round bias > $1/2^{75}$ and no 8 round bias > $1/2^{150}$ exists

AES can easily resist differential and linear cryptanalysis
Attack on 4 Rounds of AES

Plaintext Block

Secret Key

4 Operations
- Byte Substitution
- Shift Rows
- Mix Columns
- Add Round Key

Ciphertext Block

4 Operations
- Byte Substitution
- Shift Rows
- Mix Columns (except for the last round)
- Add Round Key

Key Expansion

RK1
RK2
RK3
RK4
Square Attack
(known by the AES designers)

• Works for 4 round of AES
• Can be extended up to 6 rounds
• Consider 256 plaintext blocks having the following properties
  1. byte 0 is different for in all cases (i.e. \( p_{i,0} \neq p_{j,0} \)), for \( i, j = 0 \) to 255 and \( i \neq j \)
  2. bytes 1 to 15 are the same (i.e. \( p_{i,k} = p_{j,k} \)), for \( i, j = 0 \) to 255 and \( 1 \leq k \leq 15 \)
Square Attack

• Consider 256 plaintext blocks having the following properties
  1. byte 0 is different in all cases (i.e. $p_{i,0} \neq p_{j,0}$), for $i, j = 0$ to 255 and $i \neq j$
  2. bytes 1 to 15 are the same (i.e. $p_{i,k} = p_{j,k}$), for $i, j = 0$ to 255 and $1 \leq k \leq 15$

Two properties

For some $k; 1 \leq k \leq 15$

The state is balanced

$$\bigoplus_{i=0}^{255} p_{i,k} = 0$$

$$\bigoplus_{i=0}^{255} p_{i,0} = 0$$
Square Attack
(Propagation in 3 rounds)

Add Whitening Key

Round 1
Subs Bytes
Shift Rows
Mix Columns
Add Round Key

Round 2
Sub Bytes
Shift Rows
Mix Columns
Add Round Key

Round 3
Sub Bytes
Shift Rows
Mix Columns

Active byte property
\[ \bigoplus_{i=0}^{255} p_{i,0} = 0 \]

Balanced retained
Square Attack
(Propagation in 3 rounds)

Add Whitening Key

Round 1
Subs Bytes | Shift Rows | Mix Columns | Add Round Key

Round 2
Sub Bytes | Shift Rows | Mix Columns | Add Round Key

Round 3
Sub Bytes | Shift Rows | Mix Columns | Add Round Key

Active byte property
\[ \bigoplus_{i=0}^{255} p_{i,0} = 0 \]

This property does not hold after Sub Bytes in the 4\textsuperscript{th} Round

\( s_{3,i} \quad (0 \leq i \leq 15) \)
A 4 round square attack

Round 3
Sub Bytes

Shift Rows

Mix Columns

Add Round Key

Round 4
Sub Bytes

Shift Rows

Mix Columns

Add Round Key

\[
S^{-1}(E(c_0 \oplus k_0) \oplus B(c_1 \oplus k_1) \oplus D(c_2 \oplus k_2) \oplus 9(c_3 \oplus k_3))
\]

\((c_i \oplus k_i) \text{ for } 0 \leq i \leq 3\)
4 round square attack  
(A chosen plaintext attack)

1. Choose 256 plaintexts with one active byte
2. Perform 4 round encryption for each plaintext
3. For each potential key $k_0 \parallel k_1 \parallel k_2 \parallel k_3$ do the following,
   a. Compute $s_{3,0}$ corresponding to each $c_i$ (there are 256 such $c_i$)
      call them $s_{3,0}^{(0)}, s_{3,0}^{(1)}, s_{3,0}^{(2)}, \ldots, s_{3,0}^{(255)}$
   b. compute $\bigoplus_{i=0}^{255} s_{3,0}^{(i)}$

If this is 0, then guessed $(k_0 \parallel k_1 \parallel k_2 \parallel k_3)$ may be correct
If not, guessed key is incorrect
Why square attack may lead to an incorrect key

- If the key guess is wrong, $\bigoplus_{i=0}^{255} s_{3,0}^{(i)}$ may still be 0.
- This is because $\bigoplus_{i=0}^{255} s_{3,0}^{(i)}$ evaluated to one of \{0, 1, 2, 3, ..., 255\} with equal probability

- Thus with probability $2^{-8}$, we may get $\bigoplus_{i=0}^{255} s_{3,0}^{(i)} = 0$ for the wrong key.
Extending beyond 4 rounds

Read how the square attack can be extended to 5 rounds and 6 rounds.
Related Key Attacks on AES
(theoretical attacks on full AES)

• By Alex Biryukov and Dmitry Khovratovich (2009)
• Strong assumption: the attacker forces the victim to choose keys of particular form.
• Determine how key differences affect the cipher text difference
Tracing key differences