Symmetric Ciphers

• block ciphers process messages in blocks, each of which is then en/decrypted
• like a substitution on very big characters
  – 64-bits or more
• stream ciphers process messages a bit or byte at a time when en/decrypting
• many current ciphers are block ciphers
  – better analyzed
  – broader range of applications
Block vs Stream Ciphers

(a) Stream Cipher Using Algorithmic Bit Stream Generator

(b) Block Cipher
Block Cipher Principles

• most symmetric block ciphers are based on a Feistel Cipher Structure

• needed since must be able to decrypt ciphertext to recover messages efficiently

• block ciphers look like an extremely large substitution

• would need table of $2^{64}$ entries for a 64-bit block

• instead create from smaller building blocks

• using idea of a product cipher
Claude Shannon and Substitution-Permutation Ciphers

- Claude Shannon introduced idea of substitution-permutation (S-P) networks in 1949 paper
- form basis of modern block ciphers
- S-P nets are based on the two primitive cryptographic operations seen before:
  - substitution (S-box)
  - permutation (P-box)
- provide confusion & diffusion of message & key
Confusion and Diffusion

• cipher needs to completely obscure statistical properties of original message
• a one-time pad does this
• more practically Shannon suggested combining S & P elements to obtain:
  – **diffusion** – dissipates statistical structure of plaintext over bulk of ciphertext
  – **confusion** – makes relationship between ciphertext and key as complex as possible
Feistel Cipher Structure

• Horst Feistel devised the **feistel cipher**
  – based on concept of invertible product cipher

• partitions input block into two halves
  – process through multiple rounds which
  – perform a substitution on left data half
  – based on round function of right half & subkey
  – then have permutation swapping halves

• implements Shannon’s S-P net concept
Feistel Cipher Structure
Feistel Cipher Design Elements

- block size
- key size
- number of rounds
- subkey generation algorithm
- round function
- fast software en/decryption
- ease of analysis
Data Encryption Standard (DES)

- most widely used block cipher in world
- adopted in 1977 by NBS (now NIST)
  - as FIPS PUB 46
- encrypts 64-bit data using 56-bit key
- has widespread use
- has been considerable controversy over its security
DES History

• IBM developed Lucifer cipher
  – by team led by Feistel in late 60’s
  – used 64-bit data blocks with 128-bit key
• then redeveloped as a commercial cipher with input from NSA and others
• in 1973 NBS issued request for proposals for a national cipher standard
• IBM submitted their revised Lucifer which was eventually accepted as the DES
DES Design Controversy

• although DES standard is public
• was considerable controversy over design
  – in choice of 56-bit key (vs Lucifer 128-bit)
  – and because design criteria were classified
• subsequent events and public analysis show in fact design was appropriate
• use of DES has flourished
  – especially in financial applications
  – still standardised for legacy application use
DES Encryption Overview

64-bit plaintext

Initial Permutation

Round 1

$K_1$ 48

Permutated Choice 2

Left circular shift

Round 2

$K_2$ 48

Permutated Choice 2

Left circular shift

Round 16

$K_{16}$ 48

Permutated Choice 2

Left circular shift

32-bit Swap

Inverse Initial Permutation

64-bit ciphertext
Initial Permutation IP

- first step of the data computation
- IP reorders the input data bits
- even bits to LH half, odd bits to RH half
- quite regular in structure (easy in h/w)
- example:

\[
\text{IP}(675a6967 \ 5e5a6b5a) = (ffb2194d \ 004df6fb)
\]
DES Round Structure

• uses two 32-bit L & R halves
• as for any Feistel cipher can describe as:
  
  \[ L_i = R_{i-1} \]
  
  \[ R_i = L_{i-1} \oplus F(R_{i-1}, K_i) \]

• F takes 32-bit R half and 48-bit subkey:
  – expands R to 48-bits using perm E
  – adds to subkey using XOR
  – passes through 8 S-boxes to get 32-bit result
  – finally permutes using 32-bit perm P
DES Round Structure
Substitution Boxes S

- have eight S-boxes which map 6 to 4 bits
- each S-box is actually 4 little 4 bit boxes
  - outer bits 1 & 6 (row bits) select one row of 4
  - inner bits 2-5 (col bits) are substituted
  - result is 8 lots of 4 bits, or 32 bits
DES Key Schedule

- forms subkeys used in each round
  - initial permutation of the key (PC1) which selects 56-bits in two 28-bit halves
  - 16 stages consisting of:
    - rotating each half separately either 1 or 2 places depending on the key rotation schedule $K$
    - selecting 24-bits from each half & permuting them by PC2 for use in round function F

- note practical use issues in h/w vs s/w
DES Decryption

- decrypt must unwind steps of data computation
- with Feistel design, do encryption steps again using subkeys in reverse order (SK16 ... SK1)
  - IP undoes final FP step of encryption
  - 1st round with SK16 undoes 16th encrypt round
    - ....
  - 16th round with SK1 undoes 1st encrypt round
  - then final FP undoes initial encryption IP
  - thus recovering original data value
### DES Example

<table>
<thead>
<tr>
<th>Round</th>
<th>$K_i$</th>
<th>$L_i$</th>
<th>$R_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP</td>
<td></td>
<td>5a005a00</td>
<td>3cf03c0f</td>
</tr>
<tr>
<td>1</td>
<td>1e030f03080d2930</td>
<td>3cf03c0f</td>
<td>bad22845</td>
</tr>
<tr>
<td>2</td>
<td>0a31293432242318</td>
<td>bad22845</td>
<td>99e9b723</td>
</tr>
<tr>
<td>3</td>
<td>23072318201d0c1d</td>
<td>99e9b723</td>
<td>0bae3b9e</td>
</tr>
<tr>
<td>4</td>
<td>05261d3824311a20</td>
<td>0bae3b9e</td>
<td>42415649</td>
</tr>
<tr>
<td>5</td>
<td>3325340136002c25</td>
<td>42415649</td>
<td>18b3fa41</td>
</tr>
<tr>
<td>6</td>
<td>123a2d0d04262a1c</td>
<td>18b3fa41</td>
<td>9616fe23</td>
</tr>
<tr>
<td>7</td>
<td>021f120b1c130611</td>
<td>9616fe23</td>
<td>67117cf2</td>
</tr>
<tr>
<td>8</td>
<td>1c10372a2832002b</td>
<td>67117cf2</td>
<td>c11bfc09</td>
</tr>
<tr>
<td>9</td>
<td>04292a380c341f03</td>
<td>c11bfc09</td>
<td>887fbc6c</td>
</tr>
<tr>
<td>10</td>
<td>2703212607280403</td>
<td>887fbc6c</td>
<td>600f7e8b</td>
</tr>
<tr>
<td>11</td>
<td>2826390c31261504</td>
<td>600f7e8b</td>
<td>f596506e</td>
</tr>
<tr>
<td>12</td>
<td>12071c241a0a0f08</td>
<td>f596506e</td>
<td>738538b8</td>
</tr>
<tr>
<td>13</td>
<td>300935393c0d100b</td>
<td>738538b8</td>
<td>c6a62c4e</td>
</tr>
<tr>
<td>14</td>
<td>311e09231321182a</td>
<td>c6a62c4e</td>
<td>56b0bd75</td>
</tr>
<tr>
<td>15</td>
<td>283d3e0227072528</td>
<td>56b0bd75</td>
<td>75e8fd8f</td>
</tr>
<tr>
<td>16</td>
<td>2921080b13143025</td>
<td>75e8fd8f</td>
<td>25896490</td>
</tr>
<tr>
<td>IP$^{-1}$</td>
<td></td>
<td>da02ce3a</td>
<td>89ecac3b</td>
</tr>
</tbody>
</table>
Avalanche Effect

- key desirable property of encryption alg
- where a change of one input or key bit results in changing approx half output bits
- making attempts to “home-in” by guessing keys impossible
- DES exhibits strong avalanche
Avalanche in DES

<table>
<thead>
<tr>
<th>Round</th>
<th>Value 1</th>
<th>Value 2</th>
<th>Round</th>
<th>Value 1</th>
<th>Value 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>02468aceeca86420</td>
<td>12468aceeca86420</td>
<td>9</td>
<td>c11bfc09887fbc6c</td>
<td>99f911532eed7d94</td>
</tr>
<tr>
<td>1</td>
<td>3cf03c0fbd22845</td>
<td>3cf03c0fbd22845</td>
<td>10</td>
<td>887fbc6c600f7e8b</td>
<td>2eed7d94d0f23094</td>
</tr>
<tr>
<td>2</td>
<td>bad2884599e9b723</td>
<td>bad3284539a9b7a3</td>
<td>11</td>
<td>600f7e8bf596506e</td>
<td>d0f23094455da9c4</td>
</tr>
<tr>
<td>3</td>
<td>99e9b7230bae3b9e</td>
<td>39a9b7a3171cb8b3</td>
<td>12</td>
<td>f596506e738538b8</td>
<td>455da9c47f6e3cf3</td>
</tr>
<tr>
<td>4</td>
<td>0b9e3b9e42415649</td>
<td>171cb8b3ccaca55e</td>
<td>13</td>
<td>738538b8c6a62c4e</td>
<td>7f6e3cf34bc1a8d9</td>
</tr>
<tr>
<td>5</td>
<td>424156491b3fa41</td>
<td>ccaca55ed16c3653</td>
<td>14</td>
<td>c6a62c4e56b0bd75</td>
<td>4bc1a891e07d409</td>
</tr>
<tr>
<td>6</td>
<td>18b3fa419616fe23</td>
<td>d16c3653cf402c68</td>
<td>15</td>
<td>56b0bd7575e8fd8f</td>
<td>1e07d4091ce2e6dc</td>
</tr>
<tr>
<td>7</td>
<td>9616fe2367117cf2</td>
<td>cf402c682b2cefbc</td>
<td>16</td>
<td>75e8fd8f25896490</td>
<td>1ce2e6dc365e5f59</td>
</tr>
<tr>
<td>8</td>
<td>67117cf2c11bfc09</td>
<td>2b2cefbc99f91153</td>
<td>IP⁻¹</td>
<td>da02ce3a89ecac3b</td>
<td>057cde97d7683f2a</td>
</tr>
</tbody>
</table>
Multiple Encryption & DES

• clear a replacement for DES was needed
  – theoretical attacks that can break it
  – demonstrated exhaustive key search attacks
• AES is a new cipher alternative
• prior to this alternative was to use multiple encryption with DES implementations
• Triple-DES is the chosen form
Double-DES?

• could use 2 DES encrypts on each block
  \[ C = E_{K2}(E_{K1}(P)) \]

• issue of reduction to single stage

• and have “meet-in-the-middle” attack
  – works whenever use a cipher twice
  – since \[ X = E_{K1}(P) = D_{K2}(C) \]
  – attack by encrypting \( P \) with all keys and store
  – then decrypt \( C \) with keys and match \( X \) value
  – can show takes \( O(2^{56}) \) steps
Triple-DES with Two-Keys

• hence must use 3 encryptions
  – would seem to need 3 distinct keys
• but can use 2 keys with E-D-E sequence
  – $C = E_{K_1} \left( D_{K_2} \left( E_{K_1} \left( P \right) \right) \right)$
  – nb encrypt & decrypt equivalent in security
  – if $K_1 = K_2$ then can work with single DES
• standardized in ANSI X9.17 & ISO8732
• no current known practical attacks
  – several proposed impractical attacks might become basis of future attacks
Triple-DES with Three-Keys

• although are no practical attacks on two-key Triple-DES have some indications
• can use Triple-DES with Three-Keys to avoid even these
  \[ C = E_{K3} \left( D_{K2} \left( E_{K1} (P) \right) \right) \]
• has been adopted by some Internet applications, eg PGP, S/MIME
Strength of DES – Key Size

• 56-bit keys have $2^{56} = 7.2 \times 10^{16}$ values
• brute force search looks hard
• recent advances have shown is possible
  – in 1997 on Internet in a few months
  – in 1998 on dedicated h/w (EFF) in a few days
  – in 1999 above combined in 22hrs!
• still must be able to recognize plaintext
• must now consider alternatives to DES
Strength of DES – Analytic Attacks

- now have several analytic attacks on DES
- these utilise some deep structure of the cipher
  - by gathering information about encryptions
  - can eventually recover some/all of the sub-key bits
  - if necessary then exhaustively search for the rest
- generally these are statistical attacks
  - differential cryptanalysis
  - linear cryptanalysis
  - related key attacks
Strength of DES – Timing Attacks

- attacks actual implementation of cipher
- use knowledge of consequences of implementation to derive information about some/all subkey bits
- specifically use fact that calculations can take varying times depending on the value of the inputs to it
- particularly problematic on smartcards
Block Cipher Design

- basic principles still like Feistel’s in 1970’s
- number of rounds
  - more is better, exhaustive search best attack
- function f:
  - provides “confusion”, is nonlinear, avalanche
  - have issues of how S-boxes are selected
- key schedule
  - complex subkey creation, key avalanche
Advanced Encryption Standard

• clear a replacement for DES was needed
  – have theoretical attacks that can break it
  – have demonstrated exhaustive key search attacks
• can use Triple-DES – but slow, has small blocks
• US NIST issued call for ciphers in 1997
• 15 candidates accepted in Jun 98
• 5 were shortlisted in Aug-99
• Rijndael was selected as the AES in Oct-2000
• issued as FIPS PUB 197 standard in Nov-2001
The AES Cipher - Rijndael

- designed by Rijmen-Daemen in Belgium
- has 128/192/256 bit keys, 128 bit data
- an **iterative** rather than **feistel** cipher
  - processes data as block of 4 columns of 4 bytes
  - operates on entire data block in every round
- designed to be:
  - resistant against known attacks
  - speed and code compactness on many CPUs
  - design simplicity
AES Structure

- data block of 4 columns of 4 bytes is state
- key is expanded to array of words
- has 9/11/13 rounds in which state undergoes:
  - byte substitution (1 S-box used on every byte)
  - shift rows (permute bytes between groups/columns)
  - mix columns (subs using matrix multiply of groups)
  - add round key (XOR state with key material)
  - view as alternating XOR key & scramble data bytes
- initial XOR key material & incomplete last round
- with fast XOR & table lookup implementation
Some Comments on AES

1. an **iterative** rather than **feistel** cipher
2. key expanded into array of 32-bit words
   1. four words form round key in each round
3. 4 different stages are used as shown
4. has a simple structure
5. only AddRoundKey uses key
6. AddRoundKey a form of Vernam cipher
7. each stage is easily reversible
8. decryption uses keys in reverse order
9. decryption does recover plaintext
10. final round has only 3 stages
Substitute Bytes

• a simple substitution of each byte
• uses one table of 16x16 bytes containing a permutation of all 256 8-bit values
• each byte of state is replaced by byte indexed by row (left 4-bits) & column (right 4-bits)
  – eg. byte {95} is replaced by byte in row 9 column 5
  – which has value {2A}
• S-box constructed using defined transformation of values in GF(2^8)
• designed to be resistant to all known attacks
Substitute Bytes
Substitute Bytes Example

<table>
<thead>
<tr>
<th>EA</th>
<th>04</th>
<th>65</th>
<th>85</th>
</tr>
</thead>
<tbody>
<tr>
<td>83</td>
<td>45</td>
<td>5D</td>
<td>96</td>
</tr>
<tr>
<td>5C</td>
<td>33</td>
<td>98</td>
<td>B0</td>
</tr>
<tr>
<td>F0</td>
<td>2D</td>
<td>AD</td>
<td>C5</td>
</tr>
</tbody>
</table>

→

<table>
<thead>
<tr>
<th>87</th>
<th>F2</th>
<th>4D</th>
<th>97</th>
</tr>
</thead>
<tbody>
<tr>
<td>EC</td>
<td>6E</td>
<td>4C</td>
<td>90</td>
</tr>
<tr>
<td>4A</td>
<td>C3</td>
<td>46</td>
<td>E7</td>
</tr>
<tr>
<td>8C</td>
<td>D8</td>
<td>95</td>
<td>A6</td>
</tr>
</tbody>
</table>
Shift Rows

• a circular byte shift in each each
  – 1\textsuperscript{st} row is unchanged
  – 2\textsuperscript{nd} row does 1 byte circular shift to left
  – 3rd row does 2 byte circular shift to left
  – 4th row does 3 byte circular shift to left
• decrypt inverts using shifts to right
• since state is processed by columns, this step permutes bytes between the columns
Shift Rows
Mix Columns

• each column is processed separately
• each byte is replaced by a value dependent on all 4 bytes in the column
• effectively a matrix multiplication in GF(2^8) using prime poly \( m(x) = x^8 + x^4 + x^3 + x + 1 \)

\[
\begin{bmatrix}
02 & 03 & 01 & 01 \\
01 & 02 & 03 & 01 \\
01 & 01 & 02 & 03 \\
03 & 01 & 01 & 02 \\
\end{bmatrix}
\begin{bmatrix}
S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\
S_{1,0} & S_{1,1} & S_{1,2} & S_{1,3} \\
S_{2,0} & S_{2,1} & S_{2,2} & S_{2,3} \\
S_{3,0} & S_{3,1} & S_{3,2} & S_{3,3} \\
\end{bmatrix}
= 
\begin{bmatrix}
S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\
S_{1,0} & S_{1,1} & S_{1,2} & S_{1,3} \\
S_{2,0} & S_{2,1} & S_{2,2} & S_{2,3} \\
S_{3,0} & S_{3,1} & S_{3,2} & S_{3,3} \\
\end{bmatrix}
\]
Mix Columns
Mix Columns Example

\[
\begin{array}{cccc}
87 & F2 & 4D & 97 \\
6E & 4C & 90 & EC \\
46 & E7 & 4A & C3 \\
A6 & 8C & D8 & 95 \\
\end{array}
\quad \rightarrow \quad
\begin{array}{cccc}
47 & 40 & A3 & 4C \\
37 & D4 & 70 & 9F \\
94 & E4 & 3A & 42 \\
ED & A5 & A6 & BC \\
\end{array}
\]

\[
\begin{align*}
\{02}\cdot\{87\} \oplus \{03\}\cdot\{6E\} \oplus \{46\} \oplus \{A6\} &= \{47\} \\
\{87\} \oplus \{02\}\cdot\{6E\} \oplus \{03\}\cdot\{46\} \oplus \{A6\} &= \{37\} \\
\{87\} \oplus \{6E\} \oplus \{02\}\cdot\{46\} \oplus \{03\}\cdot\{A6\} &= \{94\} \\
\{03\}\cdot\{87\} \oplus \{6E\} \oplus \{46\} \oplus \{02\}\cdot\{A6\} &= \{ED\}
\end{align*}
\]
AES Arithmetic

• uses arithmetic in the finite field GF($2^8$)
• with irreducible polynomial
  \[ m(x) = x^8 + x^4 + x^3 + x + 1 \]
  which is (100011011) or \{11b\}
• e.g.
  \[
  \{02\} \cdot \{87\} \mod \{11b\} = (1\ 0000\ 1110) \mod \{11b\}
  = (1\ 0000\ 1110) \xor (1\ 0001\ 1011) = (0001\ 0101)
  \]
Mix Columns

• can express each col as 4 equations
  – to derive each new byte in col
• decryption requires use of inverse matrix
  – with larger coefficients, hence a little harder
• have an alternate characterisation
  – each column a 4-term polynomial
  – with coefficients in GF(2^8)
  – and polynomials multiplied modulo (x^4+1)
• coefficients based on linear code with maximal distance between codewords
Add Round Key

- XOR state with 128-bits of the round key
- again processed by column (though effectively a series of byte operations)
- inverse for decryption identical
  - since XOR own inverse, with reversed keys
- designed to be as simple as possible
  - a form of Vernam cipher on expanded key
  - requires other stages for complexity / security
Add Round Key

\[
\begin{array}{cccc}
S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\
S_{1,0} & S_{1,1} & S_{1,2} & S_{1,3} \\
S_{2,0} & S_{2,1} & S_{2,2} & S_{2,3} \\
S_{3,0} & S_{3,1} & S_{3,2} & S_{3,3} \\
\end{array}
\oplus
\begin{array}{cccc}
W_i & W_{i+1} & W_{i+2} & W_{i+3} \\
\end{array}
= \begin{array}{cccc}
S'_{0,0} & S'_{0,1} & S'_{0,2} & S'_{0,3} \\
S'_{1,0} & S'_{1,1} & S'_{1,2} & S'_{1,3} \\
S'_{2,0} & S'_{2,1} & S'_{2,2} & S'_{2,3} \\
S'_{3,0} & S'_{3,1} & S'_{3,2} & S'_{3,3} \\
\end{array}
AES Round

State

SubBytes

State

ShiftRows

State

MixColumns

State

AddRoundKey

State
AES Key Expansion

- takes 128-bit (16-byte) key and expands into array of 44/52/60 32-bit words
- start by copying key into first 4 words
- then loop creating words that depend on values in previous & 4 places back
  - in 3 of 4 cases just XOR these together
  - 1\textsuperscript{st} word in 4 has rotate + S-box + XOR round constant on previous, before XOR 4\textsuperscript{th} back
AES Key Expansion

\[
\begin{array}{cccc}
& \mathbf{k}_0 & \mathbf{k}_4 & \mathbf{k}_8 & \mathbf{k}_{12} \\
\mathbf{k}_1 & \mathbf{k}_5 & \mathbf{k}_9 & \mathbf{k}_{13} \\
\mathbf{k}_2 & \mathbf{k}_6 & \mathbf{k}_{10} & \mathbf{k}_{14} \\
\mathbf{k}_3 & \mathbf{k}_7 & \mathbf{k}_{11} & \mathbf{k}_{15} \\
\end{array}
\]
Key Expansion Rationale

• designed to resist known attacks
• design criteria included
  – knowing part key insufficient to find many more
  – invertible transformation
  – fast on wide range of CPU’s
  – use round constants to break symmetry
  – diffuse key bits into round keys
  – enough non-linearity to hinder analysis
  – simplicity of description
## AES Example

### Key Expansion

<table>
<thead>
<tr>
<th>Key Words</th>
<th>Auxiliary Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>w0 = 0f 15 71 c9</td>
<td>RotWord(w3) = 7f 67 98 af = x1</td>
</tr>
<tr>
<td>w1 = 47 d9 e8 59</td>
<td>SubWord(x1) = d2 85 46 79 = y1</td>
</tr>
<tr>
<td>w2 = 0c b7 ad</td>
<td>Rcon(1) = 01 00 00 00</td>
</tr>
<tr>
<td>w3 = af 7f 67 98</td>
<td>y1 ⊕ Rcon(1) = d3 85 46 79 = z1</td>
</tr>
<tr>
<td>w4 = w0 ⊕ z1 = dc 90 37 b0</td>
<td>RotWord(w7) = 81 15 a7 38 = x2</td>
</tr>
<tr>
<td>w5 = w4 ⊕ w1 = 9b 49 df e9</td>
<td>SubWord(x4) = 0c 59 5c 07 = y2</td>
</tr>
<tr>
<td>w6 = w5 ⊕ w2 = 97 fe 72 3f</td>
<td>Rcon(2) = 02 00 00 00</td>
</tr>
<tr>
<td>w7 = w6 ⊕ w3 = 38 81 15 a7</td>
<td>y2 ⊕ Rcon(2) = 0e 59 5c 07 = z2</td>
</tr>
<tr>
<td>w8 = w4 ⊕ z2 = d2 c9 6b b7</td>
<td>RotWord(w11) = ff d3 c6 e6 = x3</td>
</tr>
<tr>
<td>w9 = w8 ⊕ w5 = 49 80 b4 5e</td>
<td>SubWord(x2) = 16 66 b4 8e = y3</td>
</tr>
<tr>
<td>w10 = w9 ⊕ w6 = de 7e c6 61</td>
<td>Rcon(3) = 04 00 00 00</td>
</tr>
<tr>
<td>w11 = w10 ⊕ w7 = e6 ff d3 c6</td>
<td>y3 ⊕ Rcon(3) = 12 66 b4 8e = z3</td>
</tr>
<tr>
<td>w12 = w8 ⊕ z3 = c0 af df 39</td>
<td>RotWord(w15) = ae 7e c0 b1 = x4</td>
</tr>
<tr>
<td>w13 = w12 ⊕ w9 = 89 2f 6b 67</td>
<td>SubWord(x3) = e4 f3 ba c8 = y4</td>
</tr>
<tr>
<td>w14 = w13 ⊕ w10 = 57 51 ad 06</td>
<td>Rcon(4) = 08 00 00 00</td>
</tr>
<tr>
<td>w15 = w14 ⊕ w11 = b1 ae 7e c0</td>
<td>y4 ⊕ Rcon(4) = ec f3 ba c8 = 4</td>
</tr>
<tr>
<td>w16 = w12 ⊕ z4 = 2c 5c 65 f1</td>
<td>RotWord(w19) = 8c dd 50 43 = x5</td>
</tr>
<tr>
<td>w17 = w16 ⊕ w13 = a5 73 0e 96</td>
<td>SubWord(x4) = 64 c1 53 1a = y5</td>
</tr>
<tr>
<td>w18 = w17 ⊕ w14 = f2 22 a3 90</td>
<td>Rcon(5) = 10 00 00 00</td>
</tr>
<tr>
<td>w19 = w18 ⊕ w15 = 43 8c dd 50</td>
<td>y5 ⊕ Rcon(5) = 74 c1 53 1a = z5</td>
</tr>
<tr>
<td>w20 = w16 ⊕ z5 = 58 9d 36 eb</td>
<td>RotWord(w23) = 40 46 bd 4c = x6</td>
</tr>
<tr>
<td>w21 = w20 ⊕ w17 = fd ee 38 7d</td>
<td>SubWord(x5) = 09 5a 7a 29 = y6</td>
</tr>
<tr>
<td>w22 = w21 ⊕ w18 = 0f cc 9b ed</td>
<td>Rcon(6) = 20 00 00 00</td>
</tr>
<tr>
<td>w23 = w22 ⊕ w19 = 4c 40 46 bd</td>
<td>y6 ⊕ Rcon(6) = 29 5a 7a 29 = z6</td>
</tr>
<tr>
<td>w24 = w20 ⊕ z6 = 71 c7 4c c2</td>
<td>RotWord(w27) = a5 a9 ef cf = x7</td>
</tr>
<tr>
<td>w25 = w24 ⊕ w21 = 8c 29 74 bf</td>
<td>SubWord(x6) = 06 d3 df 8a = y7</td>
</tr>
<tr>
<td>w26 = w25 ⊕ w22 = 83 e5 ef 52</td>
<td>Rcon(7) = 40 00 00 00</td>
</tr>
<tr>
<td>w27 = w26 ⊕ w23 = cf a5 a9 ef</td>
<td>y7 ⊕ Rcon(7) = 46 d3 df 8a = z7</td>
</tr>
<tr>
<td>w28 = w24 ⊕ z7 = 37 14 93 48</td>
<td>RotWord(w31) = 7d a1 4a f7 = x8</td>
</tr>
<tr>
<td>w29 = w28 ⊕ w25 = bb 3d e7 f7</td>
<td>SubWord(x7) = ff 32 d6 68 = y8</td>
</tr>
<tr>
<td>w30 = w29 ⊕ w26 = 38 d8 08 a5</td>
<td>Rcon(8) = 80 00 00 00</td>
</tr>
<tr>
<td>w31 = w30 ⊕ w27 = f7 7d a1 4a</td>
<td>y8 ⊕ Rcon(8) = 7f 32 d6 68 = z8</td>
</tr>
<tr>
<td>w32 = w28 ⊕ z8 = 48 26 45 20</td>
<td>RotWord(w35) = be 0b 38 3c = x9</td>
</tr>
<tr>
<td>w33 = w32 ⊕ w29 = f3 1b a2 d7</td>
<td>SubWord(x8) = ae 2b 07 eb = y9</td>
</tr>
<tr>
<td>w34 = w33 ⊕ w30 = cb c3 aa 72</td>
<td>Rcon(9) = 1b 00 00 00</td>
</tr>
<tr>
<td>w35 = w34 ⊕ w32 = 3c be 0b 38</td>
<td>y9 ⊕ Rcon(9) = b5 2b 07 eb = z9</td>
</tr>
<tr>
<td>w36 = w32 ⊕ z9 = fd 0d 42 cb</td>
<td>RotWord(w39) = 6b 41 56 f9 = x10</td>
</tr>
<tr>
<td>w37 = w36 ⊕ w33 = 0e 16 e0 1c</td>
<td>SubWord(x9) = 7f 83 b1 99 = y10</td>
</tr>
<tr>
<td>w38 = w37 ⊕ w34 = c5 d5 4a 6e</td>
<td>Rcon(10) = 36 00 00 00</td>
</tr>
<tr>
<td>w39 = w38 ⊕ w35 = f9 6b 41 56</td>
<td>y10 ⊕ Rcon(10) = 49 83 b1 99 = z10</td>
</tr>
</tbody>
</table>
AES Example

Encryption

<table>
<thead>
<tr>
<th>Start of round</th>
<th>After SubBytes</th>
<th>After ShiftRows</th>
<th>After MixColumns</th>
<th>Round Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>01 89 fe 76</td>
<td>ab 8b 89 35</td>
<td>ab 8b 89 35</td>
<td>b9 94 57 75</td>
<td>0f 47 0c af</td>
</tr>
<tr>
<td>23 ab dc 54</td>
<td>05 40 7f f1</td>
<td>40 7f f1 05</td>
<td>e4 8e 16 51</td>
<td>15 d9 b7 7f</td>
</tr>
<tr>
<td>45 cd ba 32</td>
<td>18 3f 0f fc</td>
<td>f0 fc 18 3f</td>
<td>47 20 9a 3f</td>
<td>71 e8 ad 67</td>
</tr>
<tr>
<td>67 ef 98 10</td>
<td>e4 4e 2f c4</td>
<td>e4 4e 2f c4</td>
<td>d5 c6 f5 3b</td>
<td>c9 59 d6 98</td>
</tr>
<tr>
<td>0e ce f2 d9</td>
<td>4d 76 ba e3</td>
<td>4d 76 ba e3</td>
<td>8e 22 db 12</td>
<td>dc 9b 97 38</td>
</tr>
<tr>
<td>36 72 6b 2b</td>
<td>92 c6 9b 70</td>
<td>c6 9b 70 92</td>
<td>b2 f2 dc 92</td>
<td>90 49 fe 81</td>
</tr>
<tr>
<td>34 25 17 55</td>
<td>51 16 9b e5</td>
<td>9b e5 51 16</td>
<td>df 80 f7 c1</td>
<td>37 df 72 15</td>
</tr>
<tr>
<td>67 ef 98 10</td>
<td>9d 75 7d 9e</td>
<td>9d 75 7d 9e</td>
<td>2c d5 e1 52</td>
<td>b0 e9 3f a7</td>
</tr>
<tr>
<td>5c e3 6b f4</td>
<td>4a 7f 6b bf</td>
<td>4a 7f 6b bf</td>
<td>b1 c1 0b cc</td>
<td>6b 4e c6 d3</td>
</tr>
<tr>
<td>7b 72 a2 6d</td>
<td>21 40 3a 3c</td>
<td>40 3a 3c 21</td>
<td>b3 fa 8b 07</td>
<td>7b 5e 61 c6</td>
</tr>
<tr>
<td>b4 34 31 12</td>
<td>8d 18 c7 c9</td>
<td>c7 c9 8d 18</td>
<td>f9 1f 6a c3</td>
<td>39 67 06 c0</td>
</tr>
<tr>
<td>9a 9b f7 9f</td>
<td>b8 14 d2 22</td>
<td>22 b8 14 d2</td>
<td>1d 19 24 5c</td>
<td>c0 89 57 b1</td>
</tr>
<tr>
<td>71 18 4e 5d</td>
<td>a3 52 4a ff</td>
<td>a3 52 4a ff</td>
<td>d4 11 fe 0f</td>
<td>2c a5 f2 43</td>
</tr>
<tr>
<td>15 dc da a9</td>
<td>59 86 57 d3</td>
<td>86 57 d3 59</td>
<td>3b 44 06 73</td>
<td>5c 73 22 8c</td>
</tr>
<tr>
<td>2e 74 c7 bd</td>
<td>f7 92 c6 7a</td>
<td>c6 7a 9f 72</td>
<td>cb cb 62 37</td>
<td>65 0e a3 dd</td>
</tr>
<tr>
<td>2c 7e 22 9c</td>
<td>36 f3 93 de</td>
<td>36 f3 93 de</td>
<td>19 b7 0e 7c</td>
<td>f1 96 90 50</td>
</tr>
<tr>
<td>f8 b4 0c 4c</td>
<td>41 8d fe 29</td>
<td>41 8d fe 29</td>
<td>2a 47 c4 48</td>
<td>58 fd 0f 4c</td>
</tr>
<tr>
<td>67 37 24 ff</td>
<td>85 9a 36 16</td>
<td>9a 36 16 85</td>
<td>83 e8 18 ba</td>
<td>9d ee cc 40</td>
</tr>
<tr>
<td>ae a5 c1 ea</td>
<td>84 06 78 87</td>
<td>78 87 4e 06</td>
<td>84 18 27 23</td>
<td>38 36 9b 46</td>
</tr>
<tr>
<td>e0 21 97 bc</td>
<td>9b fd 88 65</td>
<td>65 9b fd 88</td>
<td>eb 10 0a f3</td>
<td>eb 7d ed bd</td>
</tr>
<tr>
<td>72 ba cb 04</td>
<td>40 f4 1f f2</td>
<td>40 f4 1f f2</td>
<td>7b 05 42 4a</td>
<td>71 8c 83 cf</td>
</tr>
<tr>
<td>1e 06 d4 fa</td>
<td>72 6f 48 2d</td>
<td>6f 48 2d 72</td>
<td>1e d0 20 40</td>
<td>29 ce 9a 55</td>
</tr>
<tr>
<td>b2 20 bc 65</td>
<td>37 b7 65 4d</td>
<td>65 4d 37 b7</td>
<td>94 83 18 52</td>
<td>4c 7e 49 af</td>
</tr>
<tr>
<td>00 6d e7 4e</td>
<td>63 3c 94 2f</td>
<td>2f 63 3c 94</td>
<td>94 c4 43 fb</td>
<td>c2 bf 52 ef</td>
</tr>
<tr>
<td>0a 89 c1 85</td>
<td>67 a7 78 97</td>
<td>67 a7 78 97</td>
<td>ec 1a c0 80</td>
<td>37 bb 38 f7</td>
</tr>
<tr>
<td>d9 f9 c5 e5</td>
<td>35 99 a6 d9</td>
<td>99 a6 d9 35</td>
<td>0c 50 53 c7</td>
<td>14 3d 8b 7d</td>
</tr>
<tr>
<td>d8 f7 f7 fb</td>
<td>61 68 6b 0f</td>
<td>68 0f 61 68</td>
<td>3b d7 00 ef</td>
<td>93 e7 08 a1</td>
</tr>
<tr>
<td>56 7b 11 14</td>
<td>b1 21 82 fa</td>
<td>fa 21 82 b1</td>
<td>7b 22 70 ec</td>
<td>48 f7 a5 4a</td>
</tr>
<tr>
<td>db a1 f8 77</td>
<td>b9 32 41 f5</td>
<td>b9 32 41 f5</td>
<td>b1 1a 44 17</td>
<td>48 f3 cb 3c</td>
</tr>
<tr>
<td>18 6d 8b ba</td>
<td>ad 3c 3d f4</td>
<td>3c 3d f4 3d</td>
<td>3d 2f ec 6b</td>
<td>1b 26 c3 be</td>
</tr>
<tr>
<td>a8 30 08 4e</td>
<td>c2 04 30 2f</td>
<td>30 2f c2 04</td>
<td>0a 6b 2f 42</td>
<td>45 a2 aa 0b</td>
</tr>
<tr>
<td>ff d5 77 aa</td>
<td>16 03 0e ac</td>
<td>ac 16 03 0e</td>
<td>9f 68 f3 b1</td>
<td>20 72 2e 33</td>
</tr>
<tr>
<td>f9 e9 8f 2b</td>
<td>99 1e 73 f1</td>
<td>99 1e 73 f1</td>
<td>31 30 3a 2c</td>
<td>fd 0e c5 f9</td>
</tr>
<tr>
<td>1b 34 2f 08</td>
<td>af 18 15 30</td>
<td>18 15 30 18</td>
<td>7c 71 c8 c4</td>
<td>0d 16 d5 6b</td>
</tr>
<tr>
<td>4f c9 85 49</td>
<td>84 dd 97 3b</td>
<td>97 3b 84 dd</td>
<td>46 65 48 eb</td>
<td>42 e0 4a 41</td>
</tr>
<tr>
<td>bf bf 81 89</td>
<td>08 08 0c a7</td>
<td>08 08 0c a7</td>
<td>6a 1c 31 62</td>
<td>cb 1c 6e 56</td>
</tr>
<tr>
<td>cc 3e ff 3b</td>
<td>4b b2 16 e2</td>
<td>4b b2 16 e2</td>
<td>4b 86 8a 36</td>
<td>b4 8e f3 52</td>
</tr>
<tr>
<td>a1 67 59 af</td>
<td>32 85 cb 79</td>
<td>85 cb 79 32</td>
<td>b1 cb 27 5a</td>
<td>2b 9a 13 4e</td>
</tr>
<tr>
<td>04 95 02 aa</td>
<td>f2 97 77 ac</td>
<td>77 ac f2 97</td>
<td>fb f2 f2 af</td>
<td>7f 4d 59 20</td>
</tr>
<tr>
<td>a1 00 5f 34</td>
<td>32 63 cf 18</td>
<td>18 32 63 cf</td>
<td>cc 5a 5b cf</td>
<td>86 26 18 76</td>
</tr>
<tr>
<td>ff 08 69 64</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0b 53 34 14</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>84 bf ab 8f</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4a 7c 43 b9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## AES Example

### Avalanche

<table>
<thead>
<tr>
<th>Round</th>
<th>Number of bits that differ</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>58</td>
</tr>
<tr>
<td>3</td>
<td>59</td>
</tr>
<tr>
<td>4</td>
<td>61</td>
</tr>
<tr>
<td>5</td>
<td>68</td>
</tr>
<tr>
<td>6</td>
<td>64</td>
</tr>
<tr>
<td>7</td>
<td>67</td>
</tr>
<tr>
<td>8</td>
<td>65</td>
</tr>
<tr>
<td>9</td>
<td>61</td>
</tr>
<tr>
<td>10</td>
<td>58</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Round</th>
<th>Input Values</th>
<th>Output Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0123456789abcede3fedcb9876543210</td>
<td>0023456789abcede3fedcb9876543210</td>
</tr>
<tr>
<td>1</td>
<td>0e3634aece72256b6f26174ed92b5588</td>
<td>0f3634aece72256b6f26174ed92b5588</td>
</tr>
<tr>
<td>2</td>
<td>657470750fc7ff3fc0e8e8ca4dd02a9c</td>
<td>c4a9ad90fc7ff3fc0e8e8ca4dd02a9c</td>
</tr>
<tr>
<td>3</td>
<td>5c7bb49a6b72349b05a2317ff46d1294</td>
<td>fe2aee569f7e8bb8c1f5a2bb37ef53d5</td>
</tr>
<tr>
<td>4</td>
<td>7115262448dc747e5cda47227da9bd9c</td>
<td>ec093dfb7c45343d689017507d485e62</td>
</tr>
<tr>
<td>5</td>
<td>f867ae8e3b437a5210c24c1974cfeabc</td>
<td>43efdb697244df808e8d9364ee0a6ef5</td>
</tr>
<tr>
<td>6</td>
<td>721eb200ba06206dcb4bce704fa654e</td>
<td>7b28a5d5ed643287e006c099bb375302</td>
</tr>
<tr>
<td>7</td>
<td>0ad9d85689f9f77b1c5e71185e5fb14</td>
<td>3bc2d8b6798d8ac4fe36a1d891ac81a</td>
</tr>
<tr>
<td>8</td>
<td>db18a8ff8a16d30d5f88b08d777ba4eaa</td>
<td>9f8b5452023c70280e5c4bb9e555a4b</td>
</tr>
<tr>
<td>9</td>
<td>f91b4fbbfe934c9bf8f2f85812b084989</td>
<td>20264e1126b219aef7feb39b2d6de40</td>
</tr>
<tr>
<td>10</td>
<td>cca104a13e678500ff59025f3baafa34</td>
<td>b56a0341b2290ba7dfdfbddd8578205</td>
</tr>
<tr>
<td></td>
<td>ff0b844a0853bf7c6934ab4364148fb9</td>
<td>612b89398d0600cde116227ce72433f0</td>
</tr>
</tbody>
</table>
AES Decryption

• AES decryption is not identical to encryption since steps done in reverse
• but can define an equivalent inverse cipher with steps as for encryption
  – but using inverses of each step
  – with a different key schedule
• works since result is unchanged when
  – swap byte substitution & shift rows
  – swap mix columns & add (tweaked) round key
Implementation Aspects

• can efficiently implement on 8-bit CPU
  – byte substitution works on bytes using a table of 256 entries
  – shift rows is simple byte shift
  – add round key works on byte XOR’s
  – mix columns requires matrix multiply in GF($2^8$) which works on byte values, can be simplified to use table lookups & byte XOR’s
Implementation Aspects

- can efficiently implement on 32-bit CPU
  - redefine steps to use 32-bit words
  - can precompute 4 tables of 256-words
  - then each column in each round can be computed using 4 table lookups + 4 XORs
  - at a cost of 4Kb to store tables

- designers believe this very efficient implementation was a key factor in its selection as the AES cipher
Modes of Operation

• block ciphers encrypt fixed size blocks
  – eg. DES encrypts 64-bit blocks with 56-bit key
• need some way to en/decrypt arbitrary amounts of data in practise
• NIST SP 800-38A defines 5 modes
• have block and stream modes
• to cover a wide variety of applications
• can be used with any block cipher
Electronic Codebook Book (ECB)

- message is broken into independent blocks which are encrypted
- each block is a value which is substituted, like a codebook, hence name
- each block is encoded independently of the other blocks

\[ C_i = E_K(P_i) \]

- uses: secure transmission of single values
Electronic Codebook Book (ECB)
Advantages and Limitations of ECB

➢ message repetitions may show in ciphertext
  • if aligned with message block
  • particularly with data such as graphics
  • or with messages that change very little, which become a code-book analysis problem

➢ weakness is due to the encrypted message blocks being independent

➢ main use is sending a few blocks of data
Cipher Block Chaining (CBC)

- message is broken into blocks
- linked together in encryption operation
- each previous cipher blocks is chained with current plaintext block, hence name
- use Initial Vector (IV) to start process
  \[ C_i = E_K(P_i \ XOR \ C_{i-1}) \]
  \[ C_{-1} = IV \]
- uses: bulk data encryption, authentication
Cipher
Block
Chaining
(CBC)
Message Padding

- at end of message must handle a possible last short block
  - which is not as large as blocksize of cipher
  - pad either with known non-data value (eg nulls)
  - or pad last block along with count of pad size
    - eg. [ b1 b2 b3 0 0 0 0 5]
      - means have 3 data bytes, then 5 bytes pad+count
  - this may require an extra entire block over those in message

- there are other, more esoteric modes, which avoid the need for an extra block
Advantages and Limitations of CBC

- A ciphertext block depends on all blocks before it.
- Any change to a block affects all following ciphertext blocks.
- Need Initialization Vector (IV), which must be known to sender & receiver. If sent in clear, attacker can change bits of first block, and change IV to compensate. Hence IV must either be a fixed value (as in EFTPOS) or must be sent encrypted in ECB mode before rest of message.
Stream Modes of Operation

• block modes encrypt entire block
• may need to operate on smaller units
  – real time data
• convert block cipher into stream cipher
  – cipher feedback (CFB) mode
  – output feedback (OFB) mode
  – counter (CTR) mode
• use block cipher as some form of pseudo-random number generator
Cipher FeedBack (CFB)

- message is treated as a stream of bits
- added to the output of the block cipher
- result is feed back for next stage (hence name)
- standard allows any number of bit (1, 8, 64 or 128 etc) to be feed back
  - denoted CFB-1, CFB-8, CFB-64, CFB-128 etc
- most efficient to use all bits in block (64 or 128)
  
  \[ C_i = P_i \ XOR \ E_K(C_{i-1}) \]
  
  \[ C_{-1} = IV \]

- uses: stream data encryption, authentication
s-bit Cipher FeedBack (CFB-s)
Advantages and Limitations of CFB

- appropriate when data arrives in bits/bytes
- most common stream mode
- limitation is need to stall while do block encryption after every n-bits
- note that the block cipher is used in encryption mode at both ends
- errors propagate for several blocks after the error
Output FeedBack (OFB)

- message is treated as a stream of bits
- output of cipher is added to message
- output is then feed back (hence name)
- feedback is independent of message
- can be computed in advance
  \[ O_i = E_K(O_{i-1}) \]
  \[ C_i = P_i \oplus O_i \]
  \[ O_{-1} = IV \]
- uses: stream encryption on noisy channels
Output FeedBack (OFB)
Advantages and Limitations of OFB

- needs an IV which is unique for each use
  - if ever reuse attacker can recover outputs
- bit errors do not propagate
- more vulnerable to message stream modification
- sender & receiver must remain in sync
- only use with full block feedback
  - subsequent research has shown that only full block feedback (ie CFB-64 or CFB-128) should ever be used
Counter (CTR)

- a “new” mode, though proposed early on
- similar to OFB but encrypts counter value rather than any feedback value
- must have a different key & counter value for every plaintext block (never reused)

\[ O_i = E_K(i) \]
\[ C_i = P_i \ XOR \ O_i \]

- uses: high-speed network encryptions
Counter (CTR)
Advantages and Limitations of CTR

• efficiency
  – can do parallel encryptions in h/w or s/w
  – can preprocess in advance of need
  – good for bursty high speed links

• random access to encrypted data blocks

• provable security (good as other modes)

• but must ensure never reuse key/counter values, otherwise could break (cf OFB)
Feedback Characteristics

(a) Cipher block chaining (CBC) mode
(b) Cipher feedback (CFB) mode
(c) Output feedback (OFB) mode
(d) Counter (CTR) mode
XTS-AES Mode

• new mode, for block oriented storage use
  – in IEEE Std 1619-2007
• concept of tweakable block cipher
• different requirements to transmitted data
• uses AES twice for each block
  \[ T_j = E_{K2}(i) \text{ XOR } \alpha^j \]
  \[ C_j = E_{K1}(P_j \text{ XOR } T_j) \text{ XOR } T_j \]
  where \( i \) is tweak & \( j \) is sector no
• each sector may have multiple blocks
XTS-AES Mode per block
XTS-AES
Mode
Overview
Advantages and Limitations of XTS-AES

- **efficiency**
  - can do parallel encryptions in h/w or s/w
  - random access to encrypted data blocks
- has both nonce & counter
- addresses security concerned related to stored data
Differential Cryptanalysis

- one of the most significant recent (public) advances in cryptanalysis
- known by NSA in 70's DES design
- Murphy, Biham & Shamir published in 90’s
- powerful method to analyse block ciphers
- used to analyse most current block ciphers with varying degrees of success
- DES reasonably resistant to it, Lucifer
- Chosen plaintext attack: the attacker has the capability to choose arbitrary plaintexts to be encrypted and obtain the corresponding ciphertexts.
Differential Cryptanalysis

- a statistical attack against Feistel ciphers
- uses cipher structure not previously used
- design of S-P networks has output of function \( f \) influenced by both input & key
- hence cannot trace values back through cipher without knowing value of the key
- differential cryptanalysis compares two related pairs of encryptions
Differential Cryptanalysis Compares Pairs of Encryptions

- with a known difference in the input
- searching for a known difference in output
- when same subkeys are used

\[
\Delta m_{i+1} = m_{i+1} \oplus m'_{i+1} \\
= [m_{i-1} \oplus f(m_i, K_i)] \oplus [m'_{i-1} \oplus f(m'_i, K_i)] \\
= \Delta m_{i-1} \oplus [f(m_i, K_i) \oplus f(m'_i, K_i)]
\]
Differential Cryptanalysis

- have some input difference giving some output difference with probability \( p \)
- if find instances of some higher probability input / output difference pairs occurring
- can infer subkey that was used in round
- then must iterate process over many rounds (with decreasing probabilities)
Differential Cryptanalysis

- perform attack by repeatedly encrypting plaintext pairs with known input XOR until obtain desired output XOR

- when found
  - if intermediate rounds match required XOR have a **right pair**
  - if not then have a **wrong pair**, relative ratio is S/N for attack

- can then deduce keys values for the rounds
  - right pairs suggest same key bits
  - wrong pairs give random values

- for large numbers of rounds, probability is so low that more pairs are required than exist with 64-bit inputs

- Biham and Shamir have shown how a 13-round iterated characteristic can break the full 16-round DES
Linear Cryptanalysis

• find linear approximations with prob $p \neq \frac{1}{2}$

\[ P[i_1, i_2, \ldots, i_a] \oplus C[j_1, j_2, \ldots, j_b] = K[k_1, k_2, \ldots, k_c] \]

where $i_a, j_b, k_c$ are bit locations in $P, C, K$

• gives linear equation for key bits
• get one key bit using max likelihood alg
• using a large number of trial encryptions
• effectiveness given by: $|p-1/2|$
Linear Cryptanalysis

- another recent development
- also a statistical method
- must be iterated over rounds, with decreasing probabilities
- developed by Matsui et al in early 90's
- based on finding linear approximations
- can attack DES with $2^{43}$ known plaintexts, easier but still in practise infeasible
DES Design Criteria

• as reported by Coppersmith in [COPP94]
• 7 criteria for S-boxes provide for
  – non-linearity
  – resistance to differential cryptanalysis
  – good confusion
• 3 criteria for permutation P provide for
  – increased diffusion