Secret Key Systems (block encoding)

Encrypting a small block of text (say 64 bits)

General Considerations:
Secret Key Systems

Encrypting a small block of text (say 64 bits)

General Considerations:
1. Encrypted data should look random.
   As though someone flipped a fair coin 64 times and heads means 1 and tails 0.
   Any change in one bit of output corresponds to a huge change in the input (bits are uncorrelated).
   Should be about as many 1's as 0's usually.
Secret Key Systems

Encrypting a small block of text (say 64 bits)

General Considerations:
1. Encrypted data should look random.
   As though someone flipped a fair coin 64 times and heads means 1 and tails 0.
   Any change in one bit of output corresponds to a huge change in the input (bits are uncorrelated).
   Should be about as many 1's as 0's usually.
2. Crypto algorithms try to spread the influence of each input bit to all output bits and change in one input bit should have 50% chance of changing any of the output bits (hence many rounds).
Secret Key Systems

Encrypting a small block of text (say 64 bits)

General Considerations:
1. Encrypted data should look random.
   As though someone flipped a fair coin 64 times and heads means 1 and tails 0.
   Any change in one bit of output corresponds to a huge change in the input (bits are uncorrelated).
   Should be about as many 1's as 0's usually.
2. Crypto algorithms try to spread the influence of each input bit to all output bits and change in one input bit should have 50% chance of changing any of the output bits (hence many rounds).
3. Operations should be invertable – hence xor and table lookup.
   Use of one key for both encryption and decryption.
Secret Key Systems

Encrypting a small block of text (say 64 bits)

General Considerations:

1. Encrypted data should look random.
   As though someone flipped a fair coin 64 times and heads means 1 and tails 0.
   Any change in one bit of output corresponds to a huge change in the input (bits are uncorrelated).
   Should be about as many 1's as 0's usually.

2. Crypto algorithms try to spread the influence of each input bit to all output bits and change in one input bit should have 50% chance of changing any of the output bits (hence many rounds).

3. Operations should be invertable – hence xor and table lookup.
   Use of one key for both encryption and decryption.

4. Attacks may be mitigated if they rely on operations that are not efficiently implemented in hardware yet allow normal operation to complete efficiently, even in software (e.g. permute).
Secret Key Systems - DES

IBM/NSA 1977 - 64 bit blocks, 56 bit key, 8 bits parity

64 bit input block

32 bit $L_i$  32 bit $R_i$

Mangler Function

$K_i$

Encryption Round

32 bit $L_{i+1}$  32 bit $R_{i+1}$

64 bit output block
Secret Key Systems - DES

IBM/NSA 1977 - 64 bit blocks, 56 bit key, 8 bits parity

64 bit input block

32 bit $L_i$

32 bit $R_i$

Mangler Function

32 bit $L_{i+1}$

32 bit $R_{i+1}$

64 bit output block

$\oplus$

$K^n$

Encryption Round

Decryption Round
Secret Key Systems - DES

Generating per round keys $K_1 \ K_2 \ ... \ K_{16}$ from the 56 bit Key + 8 parity bits

Key bits:

| 1...8 | 9...16 | 17...24 | 25...32 | 33...40 | 41...48 | 49...56 | 57...64 |
Secret Key Systems - DES

Generating per round keys $K_1, K_2, ..., K_{16}$ from the 56 bit Key + 8 parity bits

<table>
<thead>
<tr>
<th>Key bits:</th>
<th>1...8</th>
<th>9...16</th>
<th>17...24</th>
<th>25...32</th>
<th>33...40</th>
<th>41...48</th>
<th>49...56</th>
<th>57...64</th>
</tr>
</thead>
</table>

$C_0:$
57 49 41 33 25 17 9 1 58 50 42 34 26 18 10 2 59 51 43 35 27 19 11 3 60 52 44 36

$D_0:$
63 55 47 39 31 23 15 7 62 54 46 38 30 22 14 6 61 53 45 37 29 21 13 5 28 20 12 4
Secret Key Systems - DES

Generating per round keys $K_1, K_2, ..., K_{16}$ from the 56 bit Key + 8 parity bits

Key bits:

| 1...8 | 9...16 | 17...24 | 25...32 | 33...40 | 41...48 | 49...56 | 57...64 |

$C_0$:  
57 49 41 33 25 17 9 1 58 50 42 34 26 18 10 2 59 51 43 35 27 19 11 3 60 52 44 36

$D_0$:  
63 55 47 39 31 23 15 7 62 54 46 38 30 22 14 6 61 53 45 37 29 21 13 5 28 20 12 4

Each round: $K_i$ has 48 bits assembled in 2 halves permuted from 24 bits each of $C_i$ and $D_i$. $K_{i+1}$ is obtained by rotating $C_i$ and $D_i$ left to form $C_{i+1}$ and $D_{i+1}$ (rotation is 1 bit for rounds 1, 2, 9, 16 and 2 bits for other rounds)
Secret Key Systems - DES

Generating per round keys $K_1 \ K_2 \ ... \ K_{16}$ from the 56 bit Key + 8 parity bits

Key bits:

<table>
<thead>
<tr>
<th></th>
<th>1...8</th>
<th>9...16</th>
<th>17...24</th>
<th>25...32</th>
<th>33...40</th>
<th>41...48</th>
<th>49...56</th>
<th>57...64</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_0$</td>
<td>57 49 41 33 25 17 9 1 58 50 42 34 26 18 10 2 59 51 43 35 27 19 11 3 60 52 44 36</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$D_0$</td>
<td>63 55 47 39 31 23 15 7 62 54 46 38 30 22 14 6 61 53 45 37 29 21 13 5 28 20 12 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Each round: $K_i$ has 48 bits assembled in 2 halves permuted from 24 bits each of $C_i$ and $D_i$, $K_{i+1}$ is obtained by rotating $C_i$ and $D_i$ left to form $C_{i+1}$ and $D_{i+1}$ (rotation is 1 bit for rounds 1,2,9,16 and 2 bits for other rounds)

Permutations:
Left half $C_i$: (9,18,22,25 are missing)

14 17 11 24 1 5 3 28 15 6 21 10 23 19 12 4 26 8 16 7 27 20 13 2

Right half $D_i$: (35,38,43,54 are missing)

41 52 31 37 47 55 30 40 51 45 33 48 44 49 39 56 34 53 46 42 50 36 29 32
Secret Key Systems - DES

The Mangler function: mixes 32 bit input with 48 bit key to produce 32 bits

32 bit $R_i$

Mangler Function

$K_i$

32 bit $R_{i+1}$
Secret Key Systems - DES

The Mangler function: mixes 32 bit input with 48 bit key to produce 32 bits

1. Expansion of input bits:

32 bit $R_i$

Mangler Function

32 bit $R_{i+1}$

4 bits 4 bits 4 bits 4 bits 4 bits 4 bits 4 bits 4 bits
The Mangler function: mixes 32 bit input with 48 bit key to produce 32 bits

1. Expansion of input bits:
Secret Key Systems - DES

The Mangler function: mixes 32 bit input with 48 bit key to produce 32 bits

1. Expansion of input bits:

\[ 32 \text{ bit } R_i \]

Mangler Function

\[ 32 \text{ bit } R_{i+1} \]

\[ K_i \]

\[ R_i \text{ (32 bits)} \]

\[ K_i \text{ (48 bits)} \]

Expanded input (48 bits)
Secret Key Systems - DES

The Mangler function: mixes 32 bit input with 48 bit key to produce 32 bits

1. Expansion of input bits:

2. Mixing with key:
Secret Key Systems - DES

The Mangler function: mixes 32 bit input with 48 bit key to produce 32 bits

1. Expansion of input bits:

2. Mixing with key:
Secret Key Systems - DES

The $S$ Box: maps 6 bit blocks to 4 bit sections

$S Box_1$ (first 6 bits):

Input bits 1 and 6

Input bits 2,3,4,5

Final permutation:

16 7 20 21 29 12 28 17 1 15 23 26 5 18 31 10 2 8 24 14 32 27 3 9 19 13 30 6 22 11 4 25
Weak and semi-weak keys:
If key is such that $C_0$ or $D_0$ are: 1) all 0s; 2) all 1s;
3) alternating 1s and 0s, then attack is easy. There are
16 such keys. Keys for which $C_0$ and $D_0$ are both 0 or
both 1 are called weak (encrypting with key gives same
result as decrypting).
Secret Key Systems - DES

Weak and semi-weak keys:
If key is such that $C_0$ or $D_0$ are: 1) all 0s; 2) all 1s;
3) alternating 1s and 0s, then attack is easy. There are 16 such keys. Keys for which $C_0$ and $D_0$ are both 0 or both 1 are called weak (encrypting with key gives same result as decrypting).

Discussion:
1. Not much known about the design – not made public
   Probably attempt to prevent known attacks
2. Changing S-Boxes has resulted in provably weaker system
Secret Key Systems - 3DES

Two keys $K_1$ and $K_2$:

\[
\begin{align*}
K_1 & \quad K_2 & \quad K_1 \\
\downarrow & \quad \downarrow & \quad \downarrow \\
\rightarrow & \quad \rightarrow & \quad \rightarrow \\
m & \rightarrow \ E & \rightarrow \ D & \rightarrow \ E & \rightarrow \ c \\
\text{encryption} & \quad & \text{decryption}
\end{align*}
\]
Secret Key Systems - 3DES

Two keys $K_1$ and $K_2$:

$K_1 \quad K_2 \quad K_1$

$m \rightarrow E \rightarrow D \rightarrow E \rightarrow c$

encryption

$c \rightarrow D \rightarrow E \rightarrow D \rightarrow m$

decryption

Why not 2DES

1. Double encryption with the same key still requires searching $2^{56}$ keys
Secret Key Systems - 3DES

Two keys $K_1$ and $K_2$:

\[
\begin{array}{ccc}
K_1 & K_2 & K_1 \\
\downarrow & \downarrow & \downarrow \\
m \rightarrow E \rightarrow D \rightarrow E \rightarrow c & c \rightarrow D \rightarrow E \rightarrow D \rightarrow m
\end{array}
\]

encryption decryption

Why not 2DES

1. Double encryption with the same key still requires searching $2^{56}$ keys
2. Double encryption with two different keys is just as vulnerable as DES due to the following, assuming some $<m,c>$ pairs are known:
Secret Key Systems - 3DES

Two keys $K_1$ and $K_2$:

\[
\begin{array}{ccc}
K_1 & K_2 & K_1 \\
\downarrow & \downarrow & \downarrow \\
\end{array} \quad \begin{array}{ccc}
K_1 & K_2 & K_1 \\
\downarrow & \downarrow & \downarrow \\
\end{array}
\]

\[
\begin{array}{c}
m \rightarrow E \rightarrow D \rightarrow E \rightarrow c \\
\text{encryption} \\
\end{array} \quad \begin{array}{c}
c \rightarrow D \rightarrow E \rightarrow D \rightarrow m \\
\text{decryption} \\
\end{array}
\]

Why not 2DES

1. Double encryption with the same key still requires searching $2^{56}$ keys
2. Double encryption with two different keys is just as vulnerable as DES due to the following, assuming some $<m,c>$ pairs are known:

\[
\begin{array}{ccc}
m & K_1 & E(K_1, m) \\
\hline
101010 & 1000011 \\
... & ... \\
100010 & 0101111 \\
001011 & 0001101 \\
\end{array}
\]
Secret Key Systems - 3DES

Two keys $K_1$ and $K_2$:

\[
\begin{array}{ccc}
  K_1 & K_2 & K_1 \\
  \downarrow & \downarrow & \downarrow \\
  m & \rightarrow E & D \rightarrow E \rightarrow c
\end{array}
\quad
\begin{array}{ccc}
  K_1 & K_2 & K_1 \\
  \downarrow & \downarrow & \downarrow \\
  c & \rightarrow D & E \rightarrow D \rightarrow m
\end{array}
\]

encryption
decryption

Why not 2DES

1. Double encryption with the same key still requires searching $2^{56}$ keys
2. Double encryption with two different keys is just as vulnerable as DES due to the following, assuming some $<m,c>$ pairs are known:

\[
\begin{align*}
  m: & \quad K_1 & \quad E(K_1, m) \\
  2^{56} & \{ & 101010 & 1000011 \\
  & \cdots & \cdots & \cdots \\
  & 100010 & 0101111 \\
  & 001011 & 0001101
\end{align*}
\]

\[
\begin{align*}
  c: & \quad K_2 & \quad D(K_2, c) \\
  2^{56} & \{ & 101110 & 0001101 \\
  & \cdots & \cdots & \cdots \\
  & 001110 & 1000011 \\
  & 001011 & 0001110
\end{align*}
\]
Secret Key Systems - 3DES

Two keys $K_1$ and $K_2$:

\[
\begin{pmatrix}
K_1 & K_2 & K_1 \\
\downarrow & \downarrow & \downarrow \\
m & \rightarrow & E \rightarrow D \rightarrow E \rightarrow c & c \rightarrow D \rightarrow E \rightarrow D \rightarrow m
\end{pmatrix}
\]

encryption
decryption

Why not 2DES
1. Double encryption with the same key still requires searching $2^{56}$ keys
2. Double encryption with two different keys is just as vulnerable as DES due to the following, assuming some $<m,c>$ pairs are known:

\[
\begin{array}{c|c|c}
\hline
m: & K_1 & E(K_1,m) \\
\hline
101010 & 1000011 \\
\vdots & \vdots \\
100010 & 0101111 \\
001011 & 0001101 \\
\hline
2^{56}
\end{array}
\quad \begin{array}{c|c|c}
\hline
c: & K_2 & D(K_2,c) \\
\hline
101110 & 0001101 \\
\vdots & \vdots \\
001110 & 1000011 \\
001011 & 0001101 \\
\hline
2^{56}
\end{array}
\]
Secret Key Systems - 3DES

Two keys $K_1$ and $K_2$:

$$
\begin{array}{ccc}
K_1 & K_2 & K_1 \\
\downarrow & \downarrow & \downarrow \\
\end{array}
\quad
\begin{array}{ccc}
K_1 & K_2 & K_1 \\
\downarrow & \downarrow & \downarrow \\
\end{array}
$$

$m \rightarrow E \rightarrow D \rightarrow E \rightarrow c$

encryption

c $\rightarrow$ D $\rightarrow$ E $\rightarrow$ D $\rightarrow$ m

decryption

Why not 2DES
1. Double encryption with the same key still requires searching $2^{56}$ keys
2. Double encryption with two different keys is just as vulnerable as DES due to the following, assuming some $<m,c>$ pairs are known:

<table>
<thead>
<tr>
<th>$m$: $K_1$</th>
<th>$E(K_1,m)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>101010</td>
<td>1000011</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>100010</td>
<td>0101111</td>
</tr>
<tr>
<td>001011</td>
<td>0001101</td>
</tr>
</tbody>
</table>

$2^{56}$

$\begin{array}{c|c}
m & E(K_1,m) \\
\hline
101110 & 0001101 \\
\vdots & \vdots \\
001110 & 1000011 \\
001011 & 0001101 \\
\end{array}$

Test matches on other $<m,c>$ pairs

$\begin{array}{c|c}
c & D(K_2,c) \\
\hline
101110 & 0001101 \\
\vdots & \vdots \\
001110 & 1000011 \\
001011 & 0001101 \\
\end{array}$

$m \rightarrow E(K_1,m)$

$D(K_2,c) \leftrightarrow c$
Secret Key Systems - 3DES

Why not 2DES

2. Double encryption with two different keys is just as vulnerable as DES due to the following, assuming some \(<m,c>\) pairs are known:

How many \(<m,c>\) pairs do you need?

- \(2^{64}\) possible blocks
- \(2^{56}\) table entries
- each block has probability \(2^{-8} = 1/256\) of showing in a table
- probability a block is in both tables is \(2^{-16}\)
- average number of matches is \(2^{48}\)
- average number of matches for two \(<m,c>\) pairs is about \(2^{32}\)
- for three \(<m,c>\) pairs is about \(2^{16}\)
- for four \(<m,c>\) pairs is about \(2^0\)

3. Triple encryption with two different keys
- 112 bits of key is considered enough
- straightforward to find a triple of keys that maps a given plaintext to a given ciphertext (no known attack with 2 keys)
- EDE: use same keys to get DES, EEE: effect of permutations lost
Secret Key Systems - 3DES

CBC with 3DES:

Outside:

- $m_i$ → $c_i$ with $K_1$ → E → D
- $m_{i+1}$ → $c_{i+1}$ with $K_1$ → E → D

Inside:

- $m_i$ → $c_i$ with $K_1$ → E
- $m_{i+1}$ → $c_{i+1}$ with $K_1$ → E

⊕ represents the XOR operation.
Secret Key Systems - 3DES

CBC with 3DES:

1. On the outside – same attack as with CBC – change a block with side effect of garbling another
2. On the inside – attempt at changing a block results in all block garbled to the end of the message.
3. On the inside – use three times as much hardware to pipeline encryptions resulting in DES speeds.
4. On the outside – EDE simply is a drop-in replacement for what might have been there before.
Secret Key Systems - IDEA

ETH Zuria 1991 - 64 bit blocks, 128 bit key, 0 bits parity:

Odd Encryption Round

- Multiplication mod $2^{16}+1$. 0 is treated as $2^{16}$ (invertible)
- Addition but throw away carries

64 bit input block

16 bit $X_{i,a}$

16 bit $X_{i,b}$

16 bit $X_{i,c}$

16 bit $X_{i,d}$

$K_{i,a}$

$K_{i,a}$

$K_{i,a}$

$K_{i,a}$

64 bit output block

16 bit $X_{i+1,a}$

16 bit $X_{i+1,b}$

16 bit $X_{i+1,c}$

16 bit $X_{i+1,d}$
Secret Key Systems - IDEA

ETH Zuria 1991 - 64 bit blocks, 128 bit key, 0 bits parity:

Even Encryption Round

\[ Y_{in} = X_{i,a} \otimes X_{i,b} \]
\[ Z_{in} = X_{i,c} \otimes X_{i,d} \]
\[ Y_{out} = ((K_{i,e} \otimes Y_{in}) \oplus Z_{in}) \otimes K_{i,f} \]
\[ Z_{out} = (K_{i,e} \otimes Y_{in}) \oplus Y_{out} \]
\[ X_{i+1,a} = X_{i,a} \ominus Y_{out} \]
\[ X_{i+1,b} = X_{i,b} \ominus Y_{out} \]
\[ X_{i+1,c} = X_{i,c} \ominus Z_{out} \]
\[ X_{i+1,d} = X_{i,d} \ominus Z_{out} \]
Secret Key Systems - IDEA

Key generation  52 16 bit keys needed (2 per even round 4 per odd):

128 bit key

\[ K_{1,a} \quad K_{1,b} \quad K_{1,c} \quad K_{1,d} \quad K_{2,e} \quad K_{2,f} \quad K_{3,a} \quad K_{3,b} \]
Secret Key Systems - IDEA

Key generation  52 16 bit keys needed (2 per even round 4 per odd):

\[ K_{1,a} \quad K_{1,b} \quad K_{1,c} \quad K_{1,d} \quad K_{2,e} \quad K_{2,f} \quad K_{3,a} \quad K_{3,b} \]

\[ K_{3,c} \quad K_{3,d} \quad K_{4,e} \quad K_{4,f} \quad K_{5,a} \quad K_{5,b} \quad K_{5,c} \quad K_{5,d} \]

25 bits
Secret Key Systems - AES

NIST (2001) parameterized key size (128 bits to 256 bits)

$4N_b$ octet inp

$4N_k$ octet key

key expansion

round 1

$K_0$

$K_1$
Secret Key Systems - AES

1. Byte Substitution via S-box - the transformation is invertible
Secret Key Systems - AES

1. Byte Substitution via S-box - the transformation is invertable

A byte as a polynomial: \( b_7 x^7 + b_6 x^6 + b_5 x^5 + b_4 x^4 + b_3 x^3 + b_2 x^2 + b_1 x + b_0, \ b_i \in \{0,1\} \)
Secret Key Systems - AES

1. Byte Substitution via S-box - the transformation is invertable

A byte as a polynomial: \( b_7 x^7 + b_6 x^6 + b_5 x^5 + b_4 x^4 + b_3 x^3 + b_2 x^2 + b_1 x + b_0, \ b_i \in \{0,1\} \)

Addition: \((x^6 + x^4 + x^2 + x + 1) + (x^7 + x^1 + 1) = x^7 + x^6 + x^4 + x^2 \) (exclusive or)
Secret Key Systems - AES

1. Byte Substitution via S-box - the transformation is invertable

\[
\begin{array}{c|c}
\hline
a_{0,0} & a_{0,5} \\
\hline
a_{1,0} & a_{1,5} \\
\hline
a_{2,0} & a_{2,5} \\
\hline
a_{3,0} & a_{3,5} \\
\hline
\end{array}
\quad \rightarrow \quad
\begin{array}{c|c}
\hline
b_{0,0} & b_{0,5} \\
\hline
b_{1,0} & b_{1,5} \\
\hline
b_{2,0} & b_{2,5} \\
\hline
b_{3,0} & b_{3,5} \\
\hline
\end{array}
\]

A byte as a polynomial: \( b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0 \), \( b_i \in \{0,1\} \)

Addition: \((x^6 + x^4 + x^2 + x + 1) + (x^7 + x + 1) = x^7 + x^6 + x^4 + x^2 \) (exclusive or)

Multiplication: \((x^6 + x^4 + x^2 + x + 1)(x^7 + x + 1) = x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + x + 1\)
Secret Key Systems - AES

1. Byte Substitution via S-box - the transformation is invertable

\[
\begin{array}{c|c|c|c}
& a_{0,0} & a_{0,5} & b_{0,0} \\
\hline
a_{1,0} & \ldots & a_{1,5} & b_{1,0} \\
\hline
a_{2,0} & a_{2,5} & b_{2,0} & b_{2,5} \\
\hline
a_{3,0} & a_{3,5} & b_{3,0} & b_{3,5} \\
\end{array}
\]

A byte as a polynomial: \( b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x^1 + b_0, \quad b_i \in \{0,1\} \)

Addition: \((x^6 + x^4 + x^2 + x^1 + 1) + (x^7 + x^1 + 1) = x^7 + x^6 + x^4 + x^2 \) (exclusive or)

Multiplication: \((x^6 + x^4 + x^2 + x^1 + 1) \times (x^7 + x^1 + 1) = x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + x^1 + 1\)

Irreducible polynomial: \( m(x) = x^8 + x^4 + x^3 + x^1 + 1 \)
Secret Key Systems - AES

1. Byte Substitution via S-box - the transformation is invertable

A byte as a polynomial: \( b_7 x^7 + b_6 x^6 + b_5 x^5 + b_4 x^4 + b_3 x^3 + b_2 x^2 + b_1 x^1 + b_0, \quad b_i \in \{0, 1\} \)

Addition: \((x^6 + x^4 + x^2 + x^1 + 1) + (x^7 + x^1 + 1) = x^7 + x^6 + x^4 + x^2 \) (exclusive or)

Multiplication: \((x^6 + x^4 + x^2 + x^1 + 1) * (x^7 + x^1 + 1) = x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + x^1 + 1\)

Irreducible polynomial: \( m(x) = x^8 + x^4 + x^3 + x^1 + 1 \)

Multiplication mod \( m(x) \): \((x^6 + x^4 + x^2 + x^1 + 1) * (x^7 + x^1 + 1) \) mod \( m(x) = x^7 + x^6 + 1\)
Secret Key Systems - AES

1. Byte Substitution via S-box - the transformation is invertable

A byte as a polynomial: \( b_7 x^7 + b_6 x^6 + b_5 x^5 + b_4 x^4 + b_3 x^3 + b_2 x^2 + b_1 x^1 + b_0, \; b_i \in \{0,1\} \)

Addition: \((x^6 + x^4 + x^2 + x^1 + 1) + (x^7 + x^1 + 1) = x^7 + x^6 + x^4 + x^2 \) (exclusive or)

Multiplication: \((x^6 + x^4 + x^2 + x^1 + 1) \ast (x^7 + x^1 + 1) = x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + x^1 + 1\)

Irreducible polynomial: \( m(x) = x^8 + x^4 + x^3 + x^1 + 1 \)

Multiplication mod \( m(x) \): \((x^6 + x^4 + x^2 + x^1 + 1) \ast (x^7 + x^1 + 1) \mod m(x) = x^7 + x^6 + 1\)

Use Euclid's algorithm to compute, for any \( b(x) \): \( b(x) \ast a(x) + c(x) \ast m(x) = 1 \)

That is, \( b(x) \) and \( a(x) \) are inverses modulo \( m(x) \): \( b^{-1}(x) = a(x) \mod m(x) \)

This defines the S-box.
Secret Key Systems - AES

2. Row shift (cycle, left)

\[
\begin{array}{c|c|c|c}
N_b & Row & 1 & 2 \\
\hline
4 & 1 & 2 & 3 \\
6 & 1 & 2 & 3 \\
8 & 1 & 3 & 4 \\
\end{array}
\]
Secret Key Systems - AES

3. Mixed Column Transformation – constants are 8 bits now

Let \( e(x) = e_3 x^3 + e_2 x^2 + e_1 x + e_0 \), where \( e_3 = 0x03 \), \( e_2 = e_1 = 0x01 \), \( e_0 = 0x02 \)

Then \( c(x) = e(x) \cdot b(x) \mod x^4 + 1 \), multiplication obtained via \( E \cdot b \)
Secret Key Systems - AES

4. Round Key Addition (that is, exclusive or)

Number of round key bits = \text{blk}_\text{lkngth}*(\text{rnds}+1) (e.g. 128bit, 10\text{rnds} = 1408)

Taken from expansion of cipher key

Let's forget about the expansion right now
Secret Key Systems - AES

Key Schedule example for $N_b = 6$

$W_0 \quad W_1 \quad \ldots \quad W_5 \quad W_6 \quad \ldots \quad W_{11} \quad \ldots$

1\textsuperscript{st} round

2\textsuperscript{nd} round

3\textsuperscript{rd} round
Notes:

1. Many operations are table look ups so they are fast
2. Parallelism can be exploited
3. Key expansion only needs to be done one time until the key is changed
4. The S-box minimizes the correlation between input and output bits
Secret Key Systems - AES

Number of rounds

<table>
<thead>
<tr>
<th>$N_k$</th>
<th>$N_b$</th>
<th>4</th>
<th>6</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td></td>
<td>10</td>
<td>12</td>
<td>14</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>12</td>
<td>12</td>
<td>14</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>14</td>
<td>14</td>
<td>14</td>
</tr>
</tbody>
</table>