Introduction to Information Security

Lecture 3: Block and Stream Ciphers

2007. 6.

Prof. Byoundcheon Lee
sultan (at) joongbu . ac . kr

Information and Communications University
Contents

1. Feistel Network
2. DES
3. SEED
4. AES
5. Mode of operation
6. Cryptanalysis – DC, LC
7. Stream ciphers
Symmetric Encryption Model

Plaintext $M$ $\rightarrow$ Ciphertext $C$ $\rightarrow$ Plaintext $M$

$C = E_K(M)$

$D_K(C) = M$

Key $K$

Insecure Channel

Secure Channel

Cryptanalyst Adversary

Shared Secret Key

$M'$ $\rightarrow$ $K'$
Block Cipher – A Simplified View

Input Message (Plaintext) → User Key → Encryption Round keys → Encryption Function → Output Message (Ciphertext)

Encryption Round keys → Encryption Key Schedule

Decryption Round keys → Decryption Key Schedule

Decryption Function → Output Message (Plaintext)
Most Popular Symmetric Ciphers

1980: DES
1990: 3DES
2000: AES
2010: AES
2020: AES
2030: AES

American Standards:
1997: DES (56 bit key)
1999: 3DES
2001: AES contest
128, 192, 256 bit key

Other Popular Algorithms:
IDEA
RC5
Blowfish
CAST
Serpent
RC6
Twofish
Mars
Mars

American standards

Other popular algorithms
1. Feistel Network
Feistel-type Ciphers

- Feistel network
  - An elegant variant of S-P networks that could be implemented using a single algorithm for both encryption and decryption
  - It is always a permutation regardless of the form of the $F(\ )$ function
  - $F(\ )$ does not need to be invertible

Horst Feistel
Block Cipher Architecture: Feistel-type

- **Plaintext**
  - L₀ → R₀
  - L₁ → R₁
  - …
  - Lᵣ₋₁ → Rᵣ₋₁
  - Lᵣ → Rᵣ

- **Ciphertext**
  - Rᵣ → Lᵣ
  - Rᵣ₋₁ → Lᵣ₋₁
  - …
  - R₀ → L₀

- **Encryption Process**
  - L₀ ⊕ K₁ → R₀
  - L₁ ⊕ K₂ → R₁
  - …
  - Lᵣ₋₁ ⊕ Kᵣ₋₁ → Rᵣ₋₁
  - Lᵣ ⊕ Kᵣ → Rᵣ

- **Decryption Process**
  - Rᵣ ⊕ Kᵣ → Lᵣ
  - Rᵣ₋₁ ⊕ Kᵣ₋₁ → Lᵣ₋₁
  - …
  - R₀ ⊕ K₁ → L₀

- **Function F**
  - Used in each round of encryption and decryption.
Feistel-type Cipher

\[ P = L_0 \parallel R_0 \]
\[ C = R_r \parallel L_r \]
\[ L_1 = R_0 \]
\[ R_1 = L_0 \oplus F(K_1, R_0) \]
\[ L_i = R_{i-1} \]
\[ R_i = L_{i-1} \oplus F(K_i, R_{i-1}) \]
\[ L_r = R_{r-1} \]
\[ R_r = L_{r-1} \oplus F(K_r, R_{r-1}) \]
\[ C = R_r \parallel L_r \]

for \( i = 1 \) to \( r \)

\[ L_i = R_{i-1} \]
\[ R_i = L_{i-1} \oplus F(K_i, R_{i-1}) \]
\[ C = R_r \parallel L_r \]

for \( i = r - 1 \) to \( 0 \)

\[ R_i = L_{i+1} \]
\[ L_i = R_{i+1} \oplus F(K_{i+1}, R_i) \]
\[ P = L_0 \parallel R_0 \]
Design of Feistel-type Ciphers

- **Design of F-function**
  - The only non-linear part in the Feistel-type cipher
  - Need not be invertible
  - Typically uses S-boxes (Substitution boxes) for non-linearity
  - May also contain mixing (permutation) part of the S-box outputs
  - Determines the ultimate security

- **Design of Key scheduling algorithm**
  - Algorithm for deriving as many round keys as necessary from a fixed user key
  - On-the-fly vs. off-line calculation

- **Number of rounds**
  - Depends on the strength of round function (F-function)
  - A safety margin should be considered for long-term security
  - Determined through the analysis of the whole algorithm against most powerful known cryptanalysis techniques
Lucifer

- Feistel-type Block Cipher
- Developed by H. Feistel, W. Notz, and J. L. Smith at IBM Watson research lab. in the early 1970s.
- 128-bit message space
- 128-bit ciphertext space
- 128-bit key space
- 16 rounds
2. Data Encryption Standard
Data Encryption Standard (DES)

**DES - History**
- 1976 – adopted as a federal standard
- 1977 – official publication as FIPS PUB 46

**Design Criteria of DES**
- Provide a high level of security
- Completely specify and easy to understand
- Security must depend on hidden key, not algorithm
- Available to all users
- Adaptable for use in diverse applications
- Economically implementable in electronic device
- Able to be validated
- Exportable

---

*Federal Information Processing Standards*
DES Overview

Round function

Key Scheduling
DES Overview

Plaintext $M(64)$

$IP$

$LE_0(32)$ $RE_0(32)$ $K_1$

$LE_1(32)$ $RE_1(32)$ $K_2$

$LE_2(32)$ $RE_2(32)$ $K_3$

$LE_{15}(32)$ $RE_{15}(32)$ $K_{16}$

$RE_{16}(32)$ $LE_{16}(32)$

$IP^{-1}$

Ciphertext $C(64)$
### Initial Permutation and Final Permutation

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

cf.) The 58th bit of x is the first bit of IP(x)
Function $f(k_i, RE_{i-1})$

$RE_{i-1}$ (32 bits) → Expansion E → 48 bits

48 bits → $K_i$ (48 bits)

$S_1$, $S_2$, $S_3$, $S_4$, $S_5$, $S_6$, $S_7$, $S_8$ → S-box

$P$ → Permutation P

32 bits
## Expansion E and Permutation P

<table>
<thead>
<tr>
<th>Expansion $E$</th>
<th>Permutation $P$</th>
</tr>
</thead>
<tbody>
<tr>
<td>32 1 2 3 4 5</td>
<td>16 7 20 21</td>
</tr>
<tr>
<td>4 5 6 7 8 9</td>
<td>29 12 28 17</td>
</tr>
<tr>
<td>8 9 10 11 12 13</td>
<td>1 15 23 26</td>
</tr>
<tr>
<td>12 13 14 15 16 17</td>
<td>5 18 31 10</td>
</tr>
<tr>
<td>16 17 18 19 20 21</td>
<td>2 8 24 14</td>
</tr>
<tr>
<td>20 21 22 23 24 25</td>
<td>32 27 3 9</td>
</tr>
<tr>
<td>24 25 26 27 28 29</td>
<td>19 13 30 6</td>
</tr>
<tr>
<td>28 29 30 31 32 1</td>
<td>22 11 4 25</td>
</tr>
</tbody>
</table>

cf.) 32-bits are expanded into 48-bits. Some bits are selected more than once.  
32-bit $\rightarrow$ 32-bit permutation
**S-box (substitution box)**

Look-up a value from the table using:
- \( b_1 b_6 \): row
- \( b_2 b_3 b_4 b_5 \): column

**S\(_1\)-box table**

<table>
<thead>
<tr>
<th>( Sb_1 )</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>14</td>
<td>4</td>
<td>13</td>
<td>1</td>
<td>2</td>
<td>15</td>
<td>11</td>
<td>8</td>
<td>3</td>
<td>10</td>
<td>6</td>
<td>12</td>
<td>5</td>
<td>9</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>15</td>
<td>7</td>
<td>4</td>
<td>14</td>
<td>2</td>
<td>13</td>
<td>1</td>
<td>10</td>
<td>6</td>
<td>12</td>
<td>11</td>
<td>9</td>
<td>5</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>1</td>
<td>14</td>
<td>8</td>
<td>13</td>
<td>6</td>
<td>2</td>
<td>11</td>
<td>15</td>
<td>12</td>
<td>9</td>
<td>7</td>
<td>3</td>
<td>10</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>12</td>
<td>8</td>
<td>2</td>
<td>4</td>
<td>9</td>
<td>1</td>
<td>7</td>
<td>5</td>
<td>11</td>
<td>3</td>
<td>14</td>
<td>10</td>
<td>0</td>
<td>6</td>
<td>13</td>
</tr>
</tbody>
</table>

\( b_2 b_3 b_4 b_5 \): column
**DES S-Boxes**

<table>
<thead>
<tr>
<th>S(_3)-box</th>
<th>10</th>
<th>0</th>
<th>9</th>
<th>14</th>
<th>6</th>
<th>3</th>
<th>15</th>
<th>5</th>
<th>1</th>
<th>13</th>
<th>12</th>
<th>7</th>
<th>11</th>
<th>4</th>
<th>2</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>7</td>
<td>0</td>
<td>9</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td>10</td>
<td>2</td>
<td>8</td>
<td>5</td>
<td>14</td>
<td>12</td>
<td>11</td>
<td>15</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>6</td>
<td>4</td>
<td>9</td>
<td>8</td>
<td>15</td>
<td>3</td>
<td>0</td>
<td>11</td>
<td>1</td>
<td>2</td>
<td>12</td>
<td>5</td>
<td>10</td>
<td>14</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>13</td>
<td>0</td>
<td>6</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>4</td>
<td>15</td>
<td>14</td>
<td>3</td>
<td>11</td>
<td>5</td>
<td>2</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>S(_4)-box</td>
<td>7</td>
<td>13</td>
<td>14</td>
<td>3</td>
<td>0</td>
<td>6</td>
<td>9</td>
<td>10</td>
<td>1</td>
<td>2</td>
<td>8</td>
<td>5</td>
<td>11</td>
<td>12</td>
<td>4</td>
<td>15</td>
</tr>
<tr>
<td>13</td>
<td>8</td>
<td>11</td>
<td>5</td>
<td>6</td>
<td>15</td>
<td>0</td>
<td>3</td>
<td>4</td>
<td>7</td>
<td>2</td>
<td>12</td>
<td>1</td>
<td>10</td>
<td>14</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>6</td>
<td>9</td>
<td>0</td>
<td>12</td>
<td>11</td>
<td>7</td>
<td>13</td>
<td>15</td>
<td>1</td>
<td>3</td>
<td>14</td>
<td>5</td>
<td>2</td>
<td>8</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>0</td>
<td>6</td>
<td>10</td>
<td>1</td>
<td>13</td>
<td>8</td>
<td>9</td>
<td>4</td>
<td>5</td>
<td>11</td>
<td>12</td>
<td>7</td>
<td>2</td>
<td>14</td>
<td></td>
</tr>
</tbody>
</table>
**DES S-boxes**

- 8 S-boxes (6 → 4 bits)
- each row: permutation of 0-15
- 4 rows: chosen by MSB & LSB of input
- some known design criteria
  - not linear
  - Any one bit of the inputs changes at least two output bits
  - $S(x)$ and $S(x \oplus 001100)$ differs at least 2bits
  - $S(x) \neq S(x \oplus 11ef00)$ for any $ef$
  - Resistance against DC etc.
  - The actual design principles have never been revealed (US classified information)

**Exercise:** For the $S_1$-box check whether the following property holds

- $S(x)$ and $S(x \oplus 001100)$ differs at least 2bits
Key Scheduling

Key (64)

\[ C_0(28) \]

\[ LS_1 \]

\[ C_1(28) \]

\[ LS_2 \]

\[ C_2(28) \]

\[ LS_{16} \]

\[ C_{16}(28) \]

\[ D_0(28) \]

\[ LS_1 \]

\[ D_1(28) \]

\[ LS_2 \]

\[ D_{16}(28) \]

\[ PC_2 \]

K₁

\[ PC_2 \]

K₂

\[ PC_2 \]

K₁₆
\( \textbf{PC}_1 \)

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

64 bit \( \rightarrow \) 56 bit  (Actual key size of DES is 56-bit)
cf.) Do not use the parity check bits.

8 16 24 32 40 48 56 64 was removed
### $\text{PC}_2$

<table>
<thead>
<tr>
<th>14</th>
<th>17</th>
<th>11</th>
<th>24</th>
<th>1</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>28</td>
<td>15</td>
<td>6</td>
<td>21</td>
<td>10</td>
</tr>
<tr>
<td>23</td>
<td>19</td>
<td>12</td>
<td>4</td>
<td>26</td>
<td>8</td>
</tr>
<tr>
<td>16</td>
<td>7</td>
<td>27</td>
<td>20</td>
<td>13</td>
<td>2</td>
</tr>
<tr>
<td>41</td>
<td>52</td>
<td>31</td>
<td>37</td>
<td>47</td>
<td>55</td>
</tr>
<tr>
<td>30</td>
<td>40</td>
<td>51</td>
<td>45</td>
<td>33</td>
<td>48</td>
</tr>
<tr>
<td>44</td>
<td>49</td>
<td>39</td>
<td>56</td>
<td>34</td>
<td>53</td>
</tr>
<tr>
<td>46</td>
<td>42</td>
<td>50</td>
<td>36</td>
<td>29</td>
<td>32</td>
</tr>
</tbody>
</table>

56 bit -> 48 bit

$9, 18, 22, 25, 35, 38, 43, 54$ was removed
### Left Shift $LS_s$

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Shift</th>
<th>Iteration</th>
<th>Shift</th>
</tr>
</thead>
<tbody>
<tr>
<td>$LS_1$</td>
<td>1</td>
<td>$LS_9$</td>
<td>1</td>
</tr>
<tr>
<td>$LS_2$</td>
<td>1</td>
<td>$LS_{10}$</td>
<td>2</td>
</tr>
<tr>
<td>$LS_3$</td>
<td>2</td>
<td>$LS_{11}$</td>
<td>2</td>
</tr>
<tr>
<td>$LS_4$</td>
<td>2</td>
<td>$LS_{12}$</td>
<td>2</td>
</tr>
<tr>
<td>$LS_5$</td>
<td>2</td>
<td>$LS_{13}$</td>
<td>2</td>
</tr>
<tr>
<td>$LS_6$</td>
<td>2</td>
<td>$LS_{14}$</td>
<td>2</td>
</tr>
<tr>
<td>$LS_7$</td>
<td>2</td>
<td>$LS_{15}$</td>
<td>2</td>
</tr>
<tr>
<td>$LS_8$</td>
<td>2</td>
<td>$LS_{16}$</td>
<td>1</td>
</tr>
</tbody>
</table>
Data Encryption Standard (DES)

- **DES - Controversies**
  - Unknown design criteria
  - Slow in software
  - Too short key size – 56 bits

- **DES Crack Machine**
  - Can test over 90 billion keys per second
  - EFF's "Deep Crack" and the Distributed.Net computers were testing 245 billion keys per second
  - On Jan. 19, 1999, RSA DES-III Challenge was deciphered after searching 22h. and 15m.

http://www.rsa.com/rsalabs/node.asp?id=2108

Identifier: DES-Challenge-III
Cipher: DES
Start: January 18, 1999 9:00 AM PST
Prize: $10,000
IV: da 4b be f1 6b 6e 98 3d
Plaintext: See you in Rome (second AES Conference, March 22-23, 1999)
Double DES & Triple DES

- How to strengthen existing DES implementations?

- Double DES
  - Essentially no security increase: $E_{K_1}(P) = X = D_{K_2}(C)$

- Triple DES
  - Two-key 3DES: $K_1 = K_3$
3. SEED

Korean standard encryption algorithm

SEED Algorithm

Features

- Feistel structure with 16 rounds
- 128-bits input-output data block size
- 128-bits key size
- Two 8x8 S-Boxes
- Mixed operation of XOR and modular addition
SEED Architecture

\[ \text{L}_i = (\text{L}_{i,0}, \text{L}_{i,1}) \]

\[ \text{K}_i \]

\[ \text{F} \]

\[ \text{R}_i = (\text{R}_{i,0}, \text{R}_{i,1}) \]

\[ \text{32-bit} \]

\[ \text{C} \]

\[ \oplus \]

\[ \text{K}_{i,j} \]

\[ \text{D} \]

\[ \oplus \]

\[ \text{K}_{i,j} \]

\[ \text{64-bit} \]

\[ \text{L}_{15} \]

\[ \oplus \]

\[ \text{L}_{16} \]

\[ \text{64-bit} \]

\[ \text{K}_{i,0}; \text{K}_{i,1} \]

\[ \text{i번째 라운드키} \]
The G Function

32-bit

X₃ X₂ X₁ X₀

S₂ S₁ S₂ S₁

& m₃ & m₂ & m₁ & m₀ & m₃ & m₂ & m₁ & m₀ & m₃ & m₂ & m₁ & m₀ & m₃ & m₂ & m₁ & m₀
⊕ ⊕ ⊕ ⊕

32-bit

Z₃ Z₂ Z₁ Z₀

Two 8x8 S-boxes

m₀ = 11111100
m₁ = 11110011
m₂ = 11001111
m₃ = 00111111

Permutation
Two 8x8 S-boxes

\[ S_i : \mathbb{Z}_2^8 \rightarrow \mathbb{Z}_2^8, S_i(x) = A^{(i)} \cdot x^{n_i} \oplus b_i \]

\[ A^{(1)} = \begin{pmatrix}
1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 0
\end{pmatrix}, \quad A^{(2)} = \begin{pmatrix}
0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 0
\end{pmatrix}. \]
SEED Key Scheduling

A, B, C, D : 32비트

A || B
>>8

A
B

C
D

KC0
-
G
K1,0

KC1
-
G
K2,0

B - D

K1,1 = G(B - D + KC0)

K1,0 = G(A + C - KC0)

C || D
<<8

A
B

C
D

KC0
-
G
K1,1

KC1
-
G
K2,1

B - D
4. Advanced Encryption Standard

AES Contest

- **AES Contest Calendar**
  - 1997 : Call For AES Candidate Algorithms by NIST
    - Mars, Twofish, RC6, SAFER+, HPC, CAST256, DEAL, Frog, Magenta, Rijndael, DFC, Serpent, Crypton, E2, LOKI97
  - 1999 : 2nd Round Candidates – 5 Algorithms
    - MARS, RC6, Rijndael, Serpent, and Twofish
  - 2000. 10 : **Rijndael** selected as the finalist
  - 2001. 12: official publication as FIPS PUB 197

* National Institute of Standards and Technology*
## AES Contest

- **1st Round Candidates – 15 Algorithms**

<table>
<thead>
<tr>
<th>Cipher</th>
<th>Submitted by</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAST-256</td>
<td>Entrust</td>
<td>Canada</td>
</tr>
<tr>
<td>Crypton</td>
<td>Future Systems</td>
<td>Korea‡</td>
</tr>
<tr>
<td>Deal</td>
<td>Outerbridge</td>
<td>Canada†</td>
</tr>
<tr>
<td>DFC</td>
<td>ENS–CNRS</td>
<td>France</td>
</tr>
<tr>
<td>E2</td>
<td>NTT</td>
<td>Japan</td>
</tr>
<tr>
<td>Frog*</td>
<td>TecApro</td>
<td>Costa Rica</td>
</tr>
<tr>
<td>HPC*</td>
<td>Schroepel</td>
<td>USA</td>
</tr>
<tr>
<td>LOKI97*</td>
<td>Brown, Pieprzyk, Seberry</td>
<td>Australia</td>
</tr>
<tr>
<td>Magenta</td>
<td>Deutsche Telekom</td>
<td>Germany</td>
</tr>
<tr>
<td>Mars</td>
<td>IBM</td>
<td>USA†</td>
</tr>
<tr>
<td>RC6</td>
<td>RSA</td>
<td>USA†</td>
</tr>
<tr>
<td>Rijndael*</td>
<td>Daemen, Rijmen</td>
<td>Belgium†</td>
</tr>
<tr>
<td>Safer+*</td>
<td>Cylink</td>
<td>USA†</td>
</tr>
<tr>
<td>Serpent*</td>
<td>Anderson, Biham, Knudsen</td>
<td>UK, Israel, Norway</td>
</tr>
<tr>
<td>Twofish*</td>
<td>Counterpane</td>
<td>USA†</td>
</tr>
</tbody>
</table>

* Placed in the public domain; † and foreign designers; ‡ foreign influence
## AES Contest

- **2nd Round Candidates – 5 Algorithms**

<table>
<thead>
<tr>
<th>Cipher</th>
<th>Submitter</th>
<th>Structure</th>
<th>Nonlinear Component</th>
</tr>
</thead>
<tbody>
<tr>
<td>MARS</td>
<td>IBM</td>
<td>Feistel structure</td>
<td>Sbox DD-Rotation</td>
</tr>
<tr>
<td>RC6</td>
<td>RSA Lab.</td>
<td>Feistel structure</td>
<td>Rotation</td>
</tr>
<tr>
<td>Rijndael</td>
<td>Daemen, Rijmen</td>
<td>SPN structure</td>
<td>Sbox</td>
</tr>
<tr>
<td>Serpent</td>
<td>Anderson, Biham, Knudsen</td>
<td>SPN structure</td>
<td>Sbox</td>
</tr>
<tr>
<td>Twofish</td>
<td>Schneier et. al</td>
<td>Feistel structure</td>
<td>Sbox</td>
</tr>
</tbody>
</table>
AES Contest

- AES Contest
  - 2000. 10: Rijndael selected as the finalist
  - 2001. 12: official publication as FIPS PUB 197


Vincent Rijmen
Advanced Encryption Standard (AES)

- AES External Format

**Plaintext block**
- 128 bits

**AES**

**Ciphertext block**
- 128 bits

**Key**
- 128, 192, 256 bits
Block Cipher Architecture: SPN-type

Key mixing part →

Non-linear part $r$ times repetitions

Linear part

For symmetry (optional) →

Substitution Layer $K_i (1 \leq i \leq r)$

Key Addition

Diffusion Layer

Key Addition

Output Transformation

Ciphertext

Plaintext
AES Architecture

- SPN-type block cipher
- Block size = 128 bits
- Key size / No. round
  - 128 bits \(\rightarrow\) 10 rounds
  - 192 bits \(\rightarrow\) 12 rounds
  - 256 bits \(\rightarrow\) 14 rounds

- Round transformation
  - SubBytes
  - ShiftRow
  - MixColumn
  - AddRoundKey
Round Transformation

**S-box**

SubBytes

MixColumns

ShiftRows

The input block is XOR-ed with the round key

AddRoundKey
SubBytes

1. First, taking the multiplicative inverse in GF(2^8), with the representation defined in Section 2.1. ‘00’ is mapped onto itself.

2. Then, applying an affine (over GF(2)) transformation defined by:

\[
\begin{bmatrix}
    y_0 \\
    y_1 \\
    y_2 \\
    y_3 \\
    y_4 \\
    y_5 \\
    y_6 \\
    y_7
\end{bmatrix} = \begin{bmatrix}
    1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
    1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\
    1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
    1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\
    1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
    0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
    0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\
    0 & 0 & 0 & 1 & 1 & 1 & 1 & 1
\end{bmatrix}
\begin{bmatrix}
x_0 \\
x_1 \\
x_2 \\
x_3 \\
x_4 \\
x_5 \\
x_6 \\
x_7
\end{bmatrix}
+ \begin{bmatrix}
1 \\
1 \\
0 \\
0 \\
0 \\
0 \\
1 \\
0
\end{bmatrix}
\]

\[m(x) = x^8 + x^4 + x^3 + x + 1\]
S-box for Rijndael

For a 8-bit input $abcdefg$
look for the entry in the $abcd$ row and $efgh$ column

<table>
<thead>
<tr>
<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>00</td>
<td>63</td>
<td>7c</td>
<td>77</td>
<td>7b</td>
<td>f2</td>
<td>6b</td>
<td>6f</td>
<td>c5</td>
<td>30</td>
<td>01</td>
<td>67</td>
<td>2b</td>
<td>fe</td>
<td>d7</td>
<td>ab</td>
</tr>
<tr>
<td>10</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>
</tr>
<tr>
<td>20</td>
<td>b7</td>
<td>fd</td>
<td>93</td>
<td>26</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>30</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>
</tr>
<tr>
<td>40</td>
<td>09</td>
<td>83</td>
<td>2c</td>
<td>1a</td>
<td>1b</td>
<td>6e</td>
<td>5a</td>
<td>a0</td>
<td>52</td>
<td>3b</td>
<td>d6</td>
<td>b3</td>
<td>29</td>
<td>e3</td>
<td>2f</td>
</tr>
<tr>
<td>50</td>
<td>53</td>
<td>d1</td>
<td>00</td>
<td>ed</td>
<td>20</td>
<td>fc</td>
<td>b1</td>
<td>5b</td>
<td>6a</td>
<td>cb</td>
<td>be</td>
<td>39</td>
<td>4a</td>
<td>4c</td>
<td>58</td>
</tr>
<tr>
<td>60</td>
<td>d0</td>
<td>ef</td>
<td>aa</td>
<td>fb</td>
<td>43</td>
<td>4d</td>
<td>33</td>
<td>85</td>
<td>45</td>
<td>f9</td>
<td>02</td>
<td>7f</td>
<td>50</td>
<td>3c</td>
<td>9f</td>
</tr>
<tr>
<td>70</td>
<td>51</td>
<td>a3</td>
<td>40</td>
<td>8f</td>
<td>92</td>
<td>9d</td>
<td>38</td>
<td>f5</td>
<td>bc</td>
<td>b6</td>
<td>da</td>
<td>21</td>
<td>10</td>
<td>ff</td>
<td>f3</td>
</tr>
<tr>
<td>80</td>
<td>cd</td>
<td>0c</td>
<td>13</td>
<td>ec</td>
<td>5f</td>
<td>97</td>
<td>44</td>
<td>17</td>
<td>c4</td>
<td>a7</td>
<td>7e</td>
<td>3d</td>
<td>64</td>
<td>5d</td>
<td>19</td>
</tr>
<tr>
<td>90</td>
<td>60</td>
<td>81</td>
<td>4f</td>
<td>dc</td>
<td>22</td>
<td>2a</td>
<td>90</td>
<td>88</td>
<td>46</td>
<td>ee</td>
<td>b8</td>
<td>14</td>
<td>de</td>
<td>5e</td>
<td>0b</td>
</tr>
<tr>
<td>a0</td>
<td>e0</td>
<td>32</td>
<td>3a</td>
<td>0a</td>
<td>49</td>
<td>06</td>
<td>24</td>
<td>5c</td>
<td>c2</td>
<td>d3</td>
<td>ac</td>
<td>62</td>
<td>91</td>
<td>95</td>
<td>e4</td>
</tr>
<tr>
<td>b0</td>
<td>e7</td>
<td>c8</td>
<td>37</td>
<td>6d</td>
<td>8d</td>
<td>d5</td>
<td>4e</td>
<td>a9</td>
<td>6c</td>
<td>56</td>
<td>f4</td>
<td>ea</td>
<td>65</td>
<td>7a</td>
<td>ae</td>
</tr>
<tr>
<td>c0</td>
<td>ba</td>
<td>78</td>
<td>25</td>
<td>2e</td>
<td>1c</td>
<td>a6</td>
<td>b4</td>
<td>c6</td>
<td>e8</td>
<td>dd</td>
<td>74</td>
<td>1f</td>
<td>4b</td>
<td>bd</td>
<td>8b</td>
</tr>
<tr>
<td>d0</td>
<td>70</td>
<td>3e</td>
<td>b5</td>
<td>66</td>
<td>48</td>
<td>03</td>
<td>f6</td>
<td>0e</td>
<td>61</td>
<td>35</td>
<td>57</td>
<td>b9</td>
<td>86</td>
<td>c1</td>
<td>1d</td>
</tr>
<tr>
<td>e0</td>
<td>e1</td>
<td>f8</td>
<td>98</td>
<td>11</td>
<td>69</td>
<td>d9</td>
<td>8e</td>
<td>94</td>
<td>9b</td>
<td>le</td>
<td>87</td>
<td>ee</td>
<td>ce</td>
<td>55</td>
<td>28</td>
</tr>
<tr>
<td>f0</td>
<td>8c</td>
<td>a1</td>
<td>89</td>
<td>0d</td>
<td>bf</td>
<td>e6</td>
<td>42</td>
<td>68</td>
<td>41</td>
<td>99</td>
<td>2d</td>
<td>0f</td>
<td>b0</td>
<td>54</td>
<td>bb</td>
</tr>
</tbody>
</table>
MixColumns / InvMixColumns

- Constant polynomial multiplication mod $x^4+1$

\[ c(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\} \]
\[ c^{-1}(x) = \{0b\}x^3 + \{0d\}x^2 + \{09\}x + \{0e\} \]

\[ b_j(x) = c(x) \otimes a_j(x) \]
\[ b_j(x) = c^{-1}(x) \otimes a_j(x) \]

\[
\begin{bmatrix}
02 & 03 & 01 & 01 \\
01 & 02 & 03 & 01 \\
01 & 01 & 02 & 03 \\
03 & 01 & 01 & 02
\end{bmatrix}
\begin{bmatrix}
a_0,j \\
a_1,j \\
a_2,j \\
a_3,j
\end{bmatrix}
= 
\begin{bmatrix}
0e & 0b & 0d & 09 \\
09 & 0e & 0b & 0d \\
0d & 09 & 0e & 0b \\
0b & 0d & 09 & 0e
\end{bmatrix}
\begin{bmatrix}
a_0,j \\
a_1,j \\
a_2,j \\
a_3,j
\end{bmatrix}
\]

MixColumns\hspace{2cm}InvMixColumn
The input block is XOR-ed with the round key

Figure 5: In the key addition the Round Key is bitwise EXORed to the State.
Key Scheduling

128bit key

192bit key

256bit key

SubByte
Decryption Architecture

Ciphertext

AddRoundKey

InvShiftRow

InvSubByte

AddRoundKey

InvMixColumn

InvShiftRow

InvSubByte

AddRoundKey

Plaintext

Final Round Key

Repeat N-1 Rounds

Ciphertext

AddRoundKey

InvSubByte

InvShiftRow

InvMixColumn

AddRoundKey

InvSubByte

InvShiftRow

AddRoundKey

Plaintext

Repeat N-1 Rounds
AES Engine – Top-level View

- AES hardware implementation module
  - Control logic
  - Encryption/Decryption Core
  - Key Scheduler
AES Hardware Implementation
## AES Hardware Implementation

### Table 4.1 Hardware evaluation results

<table>
<thead>
<tr>
<th>Algorithm name</th>
<th>area [Gate]</th>
<th>Key setup time [ns]</th>
<th>Critical-path [ns]</th>
<th>Throughput [Mbps]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Encryption &amp; Decryption</td>
<td>Key Schedule</td>
<td>Total</td>
<td></td>
</tr>
<tr>
<td>DES</td>
<td>42,204</td>
<td>12,201</td>
<td>54,405</td>
<td>-</td>
</tr>
<tr>
<td>Triple-DES</td>
<td>124,888</td>
<td>23,207</td>
<td>148,147</td>
<td>-</td>
</tr>
<tr>
<td>MARS</td>
<td>690,654</td>
<td>2,245,096</td>
<td>2,935,754</td>
<td>1740.99</td>
</tr>
<tr>
<td>RC6</td>
<td>741,641</td>
<td>901,382</td>
<td>1,643,037</td>
<td>2112.26</td>
</tr>
<tr>
<td>Rijndael</td>
<td>518,508</td>
<td>93,708</td>
<td>612,834</td>
<td>57.39</td>
</tr>
<tr>
<td>Serpent</td>
<td>298,533</td>
<td>205,096</td>
<td>503,770</td>
<td>114.07</td>
</tr>
<tr>
<td>Twofish</td>
<td>200,165</td>
<td>231,682</td>
<td>431,857</td>
<td>16.38</td>
</tr>
</tbody>
</table>

* CMOS ASIC Implementation by Ichikawa (Mitsubishi)
Feistel vs. SPN Structures

**Feistel structure**

64-bit block
- DES/3DES
- BLOWFISH
- CAST128
- RC5

128-bit block
- SEED
- TWOFISH
- CAST256
- RC6
- MARS

**SPN structure**

64-bit block
- SAFER
- SAFER+
- IDEA

128-bit block
- AES
- CRYPTON
- SERPENT

- Fewer constraints on the round function
- More cryptanalytic experience
- Serial in nature
- Typically $E = D$ with round keys in reverse order

- More constraints on the round function: must be invertible
- Less cryptanalytic experience: a little bit new architecture
- more parallelism
- Typically $E \neq D$
Padding for Block Cipher

- **Different padding methods according to applications**
- **PKCS Padding**: general
  - Padding length: 05 05 05 05 05
- **OneAndZero Padding**: hash
  - Padding length: 80 00 00 00 00
- **Zero Padding**: CBC-MAC
  - Padding length: 00 00 00 00 00
- **No Padding**: block-size multiple Data
- **TLS Padding**: variable padding length
  - Padding length: 04 04 04 04 04
- **SSL Padding**
5. Mode of Operation
Modes of Operation – ECB Mode

Electronic Code Book Mode

- Break a message into a sequence of plaintext blocks
- Each plaintext block is encrypted (or decrypted) independently
- The same plaintext block always produces the same ciphertext block
- May not be secure; e.g., a highly structured message
- Typically used for secure transmission of single values (e.g., encryption key)
Modes of Operation – CBC Mode

Cipher Block Chaining Mode

- Each ciphertext block is affected by previous blocks
- No fixed relationship between the plaintext block and its input to the encryption function
- The same plaintext block, if repeated, produces different ciphertext blocks
- IV (Initializing Vector) must be known to both ends
- Most widely used for block encryption

C₁ = Eₖ(P₁ ⊕ IV)  \quad C₃ = Eₖ(P₃ ⊕ C₂)

P₁ = IV ⊕ Dₖ(C₁)  \quad P₃ = C₂ ⊕ Dₖ(C₃)

C₂ = Eₖ(P₂ ⊕ C₁)  \quad C₄ = Eₖ(P₄ ⊕ C₃)

P₂ = C₁ ⊕ Dₖ(C₂)  \quad P₄ = C₃ ⊕ Dₖ(C₄)
Modes of Operation – CFB Mode

- **Cipher Feedback Mode**
  - A way of using a block cipher as a stream cipher
  - A shift register of block size maintains the current state of the cipher operation, initially set to some IV
  - The value of the shift register is encrypted using key $K$ and the leftmost $j$ bits of the output is XORed with $j$-bit plaintext $P_i$ to produce $j$-bit ciphertext $C_i$
  - The value of the shift register is shifted left by $j$ bits and the $C_i$ is fed back to the rightmost $j$ bits of the shift register
  - Typically $j = 8, 16, 32, 64 \ldots$
  - Decryption function $D_K$ is never used
Modes of Operation – OFB Mode

- **Output Feedback Mode**
  - The structure is similar to that of CFB, but
    - CFB: Ciphertext is fed back to the shift register
    - OFB: Output of $E$ is fed back to the shift register
  - For security reason, only the full feedback ($j = \text{block size}$) mode is used
  - No error propagation
  - More vulnerable to a message stream modification attack
  - May useful for secure transmission over noisy channel (e.g., satellite communication)
Modes of Operation – CTS Mode

- **Ciphertext Stealing Mode**
  - Eliminates the padding requirement for block ciphers
  - The same as CBC mode, except for the encryption/decryption of the last two blocks (one complete block and the remaining partial block)
  - Adopted in H.235 as one of operating modes for block ciphers

* H.235 covers security and encryption for H.323 and other H.245 based terminals.
* H.323 covers multimedia communication on any packet network
6. Cryptanalysis
Block Cipher – Attack Scenarios

- **Attacks on encryption schemes**
  - Ciphertext only attack: only ciphertexts are given
  - Known plaintext attack: (plaintext, ciphertext) pairs are given
  - Chosen plaintext attack: (chosen plaintext, corresponding ciphertext) pairs
  - Adaptively chosen plaintext attack
  - Chosen ciphertext attack: (chosen ciphertext, corresponding plaintext) pairs
  - Adaptively chosen ciphertext attack
Cryptanalysis of Block Ciphers

- **Statistical Cryptanalysis**
  - Differential cryptanalysis (DC)
  - Linear Cryptanalysis (LC)
  - Various key schedule cryptanalysis

- **Algebraic Cryptanalysis**
  - Interpolation attacks

- **Side Channel Cryptanalysis**
  - timing attacks
  - differential fault analysis
  - differential power analysis, etc.
Cryptanalysis of Block Ciphers - DC

- **Differential Cryptanalysis**
  - E. Biham and A. Shamir: Crypto90, Crypto92
  - Chosen plaintext attack, $O(\text{Breaking DES}_{16} \sim 2^{47})$
  - Look for correlations in Round function input and output (DES: $2^{47}$)
    - high-probability differentials, impossible differentials
    - truncated differentials, higher-order differentials


\[ \Delta X = X \oplus X' \quad \text{Input difference} \]

\[ \Delta Y = Y \oplus Y' \quad \text{Output difference} \]

\[ \Delta X = \Delta Y = \Delta X' \Delta Y' \]

Statistically non-uniform probability distribution: higher prob. for some fixed pattern $\Delta X$ & $\Delta Y$
Cryptanalysis of Block Ciphers - LC

- **Linear Cryptanalysis**
  - Matsui: Eurocrypt'93, Crypto'94
  - Known Plaintext Attack, \( O(\text{Breaking DES}_{16}) \sim 2^{43} \)
  - Look for correlations between key and cipher input and output
    - linear approximation, non-linear approximation,
    - generalized I/O sums, partitioning cryptanalysis

* M. Matsui, "Linear Cryptanalysis Method for DES Cipher", Proc. of Eurocrypt'93, LNCS765, pp.386-397
Other Attacks on Block Ciphers

- **Algebraic Cryptanalysis**
  - deterministic/probabilistic interpolation attacks

- **Key Schedule Cryptanalysis**
  - Look for correlations between key changes & cipher input/output
    - equivalent keys, weak or semi-weak keys
    - related key attacks

- **Side-Channel Cryptanalysis**
  - timing attacks
  - differential fault analysis
  - differential power analysis, etc.
7. Stream Ciphers
Stream Cipher

- Plaintext stream is bit-by-bit XORed with key stream to generate ciphertext stream
- The encryption function may vary as plaintext is processed; stateful
- Encryption depends not only on the key and plaintext, but also on the current state

Advantages
- No need for padding: the length of ciphertext = the length of plaintext
- Real-time transmission: encrypt and transmit character by character

```
User Key

Key Stream Generator

Plaintext stream → Key stream → Ciphertext stream
```

Binary Additive Stream Cipher
Stream Cipher - Example

<table>
<thead>
<tr>
<th>Plaintext</th>
<th>0 1 1 0 1 0 0 1 1 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key Stream</td>
<td>⊕ 1 1 0 0 1 0 0 0 1 0</td>
</tr>
<tr>
<td>Ciphertext</td>
<td>1 0 1 0 0 0 0 1 0 1</td>
</tr>
<tr>
<td>Key Stream</td>
<td>⊕ 1 1 0 0 1 0 0 0 1 0</td>
</tr>
<tr>
<td>Plaintext</td>
<td>0 1 1 0 1 0 0 1 1 1</td>
</tr>
</tbody>
</table>

- **RC4**
- **SEAL**

- Extremely fast Encryption/Decryption; Easy to implement in HW
- **BUT**
- Vulnerable to traffic analysis (|Plaintext| = |Ciphertext|)
- Vulnerable to various attacks without integrity check
- Using two Ciphertexts from the same key stream, we can recover the XOR of the Plaintexts
  ⇒ NEVER reuse a key stream !!
General Model of Stream Ciphers: Synchronous

- The key stream is generated independently of the plaintext & of the ciphertext
- Properties
  - Synchronization requirements
  - No error propagation
- Active attacks
  - Insertion, deletion or replay of ciphertext digits – immediate loss of sync
  - Selective change of ciphertext digits
- Additional mechanisms must be used to provide message origin authentication and message integrity
General Model of Stream Cipher: Self-synchronizing

- The key stream is generated as a function of the key & a fixed number of previous ciphertext digits

Properties

- Self-synchronization
- Limited error propagation
- Diffusion of plaintext statistics

Active attacks

- Insertion, deletion or replay of ciphertext digits – more difficult to detect
- Modification of ciphertext digits – more likelihood of being detected

Additional mechanisms must be used for message origin authentication and integrity
Stream Cipher vs. Block Cipher

- **Stream Cipher**
  - Encrypt individual character (often 1 bit)
  - Have memory; stateful cipher
  - Generally extremely faster than block ciphers
  - Suitable for multimedia streaming data (audio, video)
  - Limited / No error propagation
  - Problem: Re-sync. of key stream generator state
  - Problem: insertion/deletion, replay of ciphertext digits
    → Need additional Integrity Check

- **Block Cipher**
  - Encrypt simultaneously a group of characters (64 / 128 bits)
    → Need Padding
  - Memoryless
  - Substitution Permutation Networks (SPN) or Feistel-type
  - Various modes of operation: ECB, CBC, OFB, CFB, CTS, etc…
## Symmetric Key Ciphers - Implementation

### Block Ciphers

<table>
<thead>
<tr>
<th>Cipher</th>
<th>Block Size</th>
<th>Key Size</th>
<th>Key Scheduling</th>
<th>Enc./Dec. (Mbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DES</td>
<td>64</td>
<td>56</td>
<td>3.5/3.5 µs</td>
<td>73.8/74.1</td>
</tr>
<tr>
<td>3DES</td>
<td>64</td>
<td>112/168</td>
<td>10.4/10.3 µs</td>
<td>25.1/25.2</td>
</tr>
<tr>
<td>RC5</td>
<td>64</td>
<td>0~2048(128)</td>
<td>4.4/4.4 µs</td>
<td>173.5/191.5</td>
</tr>
<tr>
<td>SEED</td>
<td>128</td>
<td>128</td>
<td>1.3/1.3 µs</td>
<td>88.2/88.8</td>
</tr>
<tr>
<td>AES</td>
<td>128</td>
<td>128/192/256</td>
<td>3.3/3.3 µs</td>
<td>130.0/135.0</td>
</tr>
<tr>
<td>IDEA</td>
<td>64</td>
<td>128</td>
<td>1.5/13.3 µs</td>
<td>57.6/57.2</td>
</tr>
<tr>
<td>Crypton</td>
<td>128</td>
<td>0~256(128)</td>
<td>0.9/1.0 µs</td>
<td>130.9/130.0</td>
</tr>
</tbody>
</table>

### Stream Ciphers

<table>
<thead>
<tr>
<th>Cipher</th>
<th>Key Setup</th>
<th>Encryption (1 MB data)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC4</td>
<td>7.9 µs</td>
<td>189.5 Mbps</td>
</tr>
<tr>
<td>SEAL</td>
<td>697.5 µs</td>
<td>327.9 Mbps</td>
</tr>
</tbody>
</table>

---

**Feistel**

**SPN**

**PIII 450MHz**

**Widows 98**

**MSVC++ 6.0**
Homework #3

- **Block Cipher Design and Implementation**

The biggest criticism on DES is that its key length (56-bit) is too short compared with current computing environment. Design your own block cipher algorithm which has 256-bit block size and 256-bit key length. In this homework I do not require any cryptanalysis of your proposed block cipher algorithm. You can study and modify any existing block cipher algorithm (like DES) and their source code freely, but never copy directly. (2DES, 3DES, 4DES are not considered a new algorithm.)

If you think you are not good at programming, you can make a temporary team of 2-3 members and do this homework.

1. Describe your design strategy or policy
2. Describe your algorithm clearly using the techniques learned in this lecture
3. Implement a C program (or use any language you prefer) which can encrypt and decrypt message in your algorithm
4. Provide a performance analysis of your algorithm and your implementation