FEAL(Fast data Encipherment ALg.)

Ex) FEAL-8
Miyaguchi(NTT), ’87

©ICU Kwangjo Kim

FEAL(II)

\[ f(R, K) \text{-function} \]

\[ S_0(X_1, X_2) = \text{Rot}_2((X_1 + X_2 + d) \mod 256) \]
where, \( d = 0 \) or \( 1 \), \( X_1 \) and \( X_2 \) : 8bit,
\[ \text{Rot}_2(Y) : 2\text{-bit left rotation of 8-bit} Y \]

\[ K_{2p-1} : \text{Left half of } f_p(a, b), K_{2p+1} : \text{right half of } f_p(a, b) \]

©ICU Kwangjo Kim
FEAL(III)

<table>
<thead>
<tr>
<th>FEAL-n</th>
<th>Attack Method</th>
<th>Data Complexity</th>
<th>Storage Complexity</th>
<th>Processing Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Known</td>
<td>Chosen</td>
<td></td>
</tr>
<tr>
<td>FEAL-4 (LC)</td>
<td></td>
<td>5</td>
<td>-</td>
<td>30Kbytes</td>
</tr>
<tr>
<td>FEAL-6 (LC)</td>
<td></td>
<td>100</td>
<td>-</td>
<td>100Kbytes</td>
</tr>
<tr>
<td>FEAL-8 (LC)</td>
<td></td>
<td>2^24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FEAL-8 (DC)</td>
<td></td>
<td>2^7 pairs</td>
<td>280Kbytes</td>
<td></td>
</tr>
<tr>
<td>FEAL-16 (DC)</td>
<td></td>
<td>2^29 pairs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FEAL-24 (DC)</td>
<td></td>
<td>2^45 pairs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FEAL-32 (DC)</td>
<td></td>
<td>2^66 pairs</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Laǐ’s Classification of “E/D-Similar” Iterated Ciphers

(I) Involution Ciphers Only

To decrypt: Reverse the Key Schedule
(II) Involution Ciphers and Involutary Permutations

To decrypt: Reverse the Key Schedule
Ex) DES, FEAL, LOKI etc

* Group Ciphers Only

To decrypt: Reverse the Key Schedule and replace each subkey by its group inverse.
(III) Group Ciphers and Involution Ciphers

To decrypt: Reverse the Key Schedule and replace each A-key by its group inverse. Ex) PES

(*) Never should be used ... equivalent to one encipherment with $Z^{(1)} \otimes Z^{(2)} \otimes \cdots \otimes Z^{(r)}$

(IV) Group Ciphers, Involution Ciphers and Involutary Permutations such that $P_{I}(a \otimes b) = P_{I}(a) \otimes P_{I}(b)$

Because $P_{I}(P_{I}(z)) = P_{I}(P_{I}(P_{I}(z))) = P_{I}(P_{I}(z))$.

There is no $P_{I}$ in the last round.

To decrypt: Reverse the Key Schedule and replace $Z_{A}^{(1)}$ and $Z_{A}^{(r+1)}$ by its group inverse and replace all other A-subkeys by the group inverses of their involutory permuted values [ $P_{I}(Z^{(1)}) = (P_{I}(Z))^{-1}$]. Ex) IDEA
IDEA (Int'l Data Enc. Al.)

Pt (64 bit) P = (X₁, X₂, X₃, X₄)
Ct (64 bit) C = (Y₁, Y₂, Y₃, Y₄)

Zᵢ (16 bit) : r-round key block

round 1
- : Xor
- : Mul. mod \(2^{16}+1\)
- : Add. mod \(2^{16}\)
- : MA structure
All lines: 16 bit

Pt (64 bit) P = (X₁, X₂, X₃, X₄)
Ct (64 bit) C = (Y₁, Y₂, Y₃, Y₄)

Zᵢ (16 bit) : r-round key block

round 1
- : Xor
- : Mul. mod \(2^{16}+1\)
- : Add. mod \(2^{16}\)
- : MA structure
All lines: 16 bit

IDEA (II)

Divide 128-bit key into 52 16-bit word
1. Divide 128-bit into 8 blocks and use 6 blocks for round key
2. 25-bit left rotate of 128-bit
3. Repeat steps (1) and (2) till 8-round

Round Key block for Encryption / Decryption

<table>
<thead>
<tr>
<th>Round</th>
<th>Encryption Key Block</th>
<th>Decryption Key Block</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Z₁⁽¹⁾ Z₂⁽¹⁾ Z₃⁽¹⁾ Z₄⁽¹⁾ Z₅⁽¹⁾ Z₆⁽¹⁾ Z₇⁽¹⁾ Z₈⁽¹⁾</td>
<td>Z₁⁽⁷⁾⁻¹ Z₂⁽⁷⁾⁻¹ Z₃⁽⁷⁾⁻¹ Z₄⁽⁷⁾⁻¹ Z₅⁽⁷⁾⁻¹ Z₆⁽⁷⁾⁻¹ Z₇⁽⁷⁾⁻¹ Z₈⁽⁷⁾⁻¹</td>
</tr>
<tr>
<td>2</td>
<td>Z₁⁽²⁾ Z₂⁽²⁾ Z₃⁽²⁾ Z₄⁽²⁾ Z₅⁽²⁾ Z₆⁽²⁾</td>
<td>Z₁⁽⁸⁾⁻¹ Z₂⁽⁸⁾⁻¹ Z₃⁽⁸⁾⁻¹ Z₄⁽⁸⁾⁻¹ Z₅⁽⁸⁾⁻¹ Z₆⁽⁸⁾⁻¹</td>
</tr>
<tr>
<td>3</td>
<td>Z₁⁽³⁾ Z₂⁽³⁾ Z₃⁽³⁾ Z₄⁽³⁾ Z₅⁽³⁾ Z₆⁽³⁾</td>
<td>Z₁⁽⁹⁾⁻¹ Z₂⁽⁹⁾⁻¹ Z₃⁽⁹⁾⁻¹ Z₄⁽⁹⁾⁻¹ Z₅⁽⁹⁾⁻¹ Z₆⁽⁹⁾⁻¹</td>
</tr>
<tr>
<td>4</td>
<td>Z₁⁽⁴⁾ Z₂⁽⁴⁾ Z₃⁽⁴⁾ Z₄⁽⁴⁾ Z₅⁽⁴⁾ Z₆⁽⁴⁾</td>
<td>Z₁⁽¹⁰⁾⁻¹ Z₂⁽¹⁰⁾⁻¹ Z₃⁽¹⁰⁾⁻¹ Z₄⁽¹⁰⁾⁻¹ Z₅⁽¹⁰⁾⁻¹ Z₆⁽¹⁰⁾⁻¹</td>
</tr>
<tr>
<td>5</td>
<td>Z₁⁽⁵⁾ Z₂⁽⁵⁾ Z₃⁽⁵⁾ Z₄⁽⁵⁾ Z₅⁽⁵⁾ Z₆⁽⁵⁾</td>
<td>Z₁⁽¹¹⁾⁻¹ Z₂⁽¹¹⁾⁻¹ Z₃⁽¹¹⁾⁻¹ Z₄⁽¹¹⁾⁻¹ Z₅⁽¹¹⁾⁻¹ Z₆⁽¹¹⁾⁻¹</td>
</tr>
<tr>
<td>6</td>
<td>Z₁⁽⁶⁾ Z₂⁽⁶⁾ Z₃⁽⁶⁾ Z₄⁽⁶⁾ Z₅⁽⁶⁾ Z₆⁽⁶⁾</td>
<td>Z₁⁽¹²⁾⁻¹ Z₂⁽¹²⁾⁻¹ Z₃⁽¹²⁾⁻¹ Z₄⁽¹²⁾⁻¹ Z₅⁽¹²⁾⁻¹ Z₆⁽¹²⁾⁻¹</td>
</tr>
<tr>
<td>7</td>
<td>Z₁⁽⁷⁾ Z₂⁽⁷⁾ Z₃⁽⁷⁾ Z₄⁽⁷⁾ Z₅⁽⁷⁾ Z₆⁽⁷⁾</td>
<td>Z₁⁽¹³⁾⁻¹ Z₂⁽¹³⁾⁻¹ Z₃⁽¹³⁾⁻¹ Z₄⁽¹³⁾⁻¹ Z₅⁽¹³⁾⁻¹ Z₆⁽¹³⁾⁻¹</td>
</tr>
<tr>
<td>8</td>
<td>Z₁⁽⁸⁾ Z₂⁽⁸⁾ Z₃⁽⁸⁾ Z₄⁽⁸⁾ Z₅⁽⁸⁾ Z₆⁽⁸⁾</td>
<td>Z₁⁽¹⁴⁾⁻¹ Z₂⁽¹⁴⁾⁻¹ Z₃⁽¹⁴⁾⁻¹ Z₄⁽¹⁴⁾⁻¹ Z₅⁽¹⁴⁾⁻¹ Z₆⁽¹⁴⁾⁻¹</td>
</tr>
<tr>
<td>Output</td>
<td>Z₁⁽⁹⁾ Z₂⁽⁹⁾ Z₃⁽⁹⁾ Z₄⁽⁹⁾</td>
<td>Z₁⁽¹⁵⁾⁻¹ Z₂⁽¹⁵⁾⁻¹ Z₃⁽¹⁵⁾⁻¹ Z₄⁽¹⁵⁾⁻¹</td>
</tr>
</tbody>
</table>

Z⁻¹ : multiplicative inverse of Zᵢ mod \(2^{16}+1\), Zᵢ : additive inverse of Zᵢ mod \(2^{16}\)
LOKI

- Brown, Pieprzyk, Seberry (Australia)
- Modify f-function and Key Schedule of DES
  - S-box: 12 bit -> 8 bit
  - Rotation: 12-bit or 13-bit bit rotation
  - Others are same as DES
- LOKI89, LOKI91
  * Name of strong and capricious god appeared in Greek myth

RC-5(I)

- Ron Rivest: Ron’s Code
- Ease for H/W & S/W Implementation
- RC5 - w/r/b
  - w: word length (16, 32)
  - r: number of round (0-255)
  - b: key byte (0-255)
  - Ex) RC5-32/12/16: 100Kb, 586PC
- To prevent weak key, user magic constant (e, π) in key scheduling
RC-5(II)

Encryption

\[ A = A + S[0] \]
\[ B = B + S[1] \]

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

\[ A = (A \oplus B) \ll B + S[2*i] \]
\[ B = (B \oplus A) \ll A + S[2*i + 1] \]

- \( S[i] \) : Round Key

1 round

: XOR

: Rotation

: addition mod \( 2^w \)

GOST

- Russia: GOvernment STandard 28147-89
- internal specification (unpublished)
- 64-bit block cipher, 32 round
- 256-bit main key & 512-bit subkey
- No Key Scheduling
- Modify f-function of DES : 4x4 k-box
**MISTY**

- Matsui, Ichigawa, Sorimachi, Tokita, Yamagishi in Mitsubishi co.
- Pt/Ct: 64-bit, Key: 128-bit, 8 Round
- Provable security against DC and LC
- By using recursive structure 30Mb/s (HP9735, PA7150-125MHz)
- To be KASUMI for 3GPP group

**Skipjack**

- Classified 64-bit block for replacing -> Internal structure published '98
- 80-bit key, 32 round
- Used for Clipper Project with Key Escrow
- ECB, CBC, 64 bitOFB, {1/8/16/32} bit CFB
Blowfish

- Bruce Schneier, et al
- XOR, Addition for 32 bit CPU
- 64bit block, 448 bit key, 8 x 32 S-box
- 16 round
- Takes long time for key scheduling

Blowfish(II)

\[ F(X) = ((S_1 + S_2 \mod 2^{32}) \oplus S_3) \oplus S_4 \mod 2^{32} \]

\( S_1, S_2, S_3, S_4 : 4 \times 8 \times 32 \) S-boxes
SEED(I)

- (Korean Data Encryption Standard)
  - 128-bit symmetric block cipher
  - developed by KISA, ETRI, ADD + a
- Reference
  - KCryptoGate (Korean cRYPTOgraphers' GATEway) http://www.cryptogate.com

SEED(II)

- Introduction
  - Symmetric Block Cipher
  - Block size, key size : 128-bit
  - Feistel structure
  - 16 rounds
- Design criteria
  - Provide security proof
  - Secure against DC, LC, Higher order DC, Related key attack
  - Faster than 3-DES in software
  - Use nonlinear function for round-key generation
SEED(III)

SEED(IV)

$Y_0 = S_2(X_0), \ Y_1 = S_2(X_2), \ Y_1 = S_2(X_1), \ Y_0 = S_1(X_0),$

$Z = (Y \& m) \oplus (Y \& m) \oplus (Y \& m) \oplus (Y \& m) \oplus (Y \& m) \oplus (Y \& m) \oplus (Y \& m) \oplus (Y \& m) \oplus (Y \& m) \oplus (Y \& m) \oplus (Y \& m) \oplus (Y \& m) \oplus (Y \& m) \oplus (Y \& m)$

$m = 0x3f, \ m = 0x3f, \ m = 0x3f, \ m = 0x3f$
**SEED(V)**

Input 128-bit key \((A,B,C,D)\)

```
for (i=1; i<=16; i++) {
    K_{i,s} \leftarrow G(A_{i}+C_{i}+K_{i});
    K_{i,s} \leftarrow G(B_{i}+D_{i}+K_{i});
    if (s=2 \land i) A \oplus B \leftarrow (A \oplus B)^{w_{i},s};
    else C \oplus D \leftarrow (C \oplus D)^{w_{i},s};
}
```

Output round keys \(K_{i,j}\)

**SEED(VI)**

Round-key generation

**Input 128-bit key \((A,B,C,D)\)**

```
for (i=1; i<=16; i++) {
    K_{i,s} \leftarrow G(A_{i}+C_{i}+K_{i});
    K_{i,s} \leftarrow G(B_{i}+D_{i}+K_{i});
    if (s=2 \land i) A \oplus B \leftarrow (A \oplus B)^{w_{i},s};
    else C \oplus D \leftarrow (C \oplus D)^{w_{i},s};
}
```

Output round keys \(K_{i,j}\)

**Round Constants**

<table>
<thead>
<tr>
<th>Round Constants</th>
<th>Round Constants</th>
</tr>
</thead>
<tbody>
<tr>
<td>(KC_{10} = 0x0e3779b69)</td>
<td>(KC_{15} = 0x3779b69c)</td>
</tr>
<tr>
<td>(KC_{1} = 0x3e6e5735)</td>
<td>(KC_{6} = 0x0e5735c)</td>
</tr>
<tr>
<td>(KC_{2} = 0x78de6e65)</td>
<td>(KC_{7} = 0xddbe6e78)</td>
</tr>
<tr>
<td>(KC_{3} = 0xf1bdc0e)</td>
<td>(KC_{8} = 0xbdc0e1b)</td>
</tr>
<tr>
<td>(KC_{4} = 0xe6f708b9)</td>
<td>(KC_{9} = 0x79b9e3)</td>
</tr>
<tr>
<td>(KC_{5} = 0x0e5735c)</td>
<td>(KC_{10} = 0xe5735c)</td>
</tr>
<tr>
<td>(KC_{6} = 0x86de6e78)</td>
<td>(KC_{11} = 0x0e6e78)</td>
</tr>
<tr>
<td>(KC_{7} = 0x1bdc0e)</td>
<td>(KC_{12} = 0x1bdc0e)</td>
</tr>
</tbody>
</table>

©ICU Kwangjo Kim
SEED(VII)

Round-key generation

A, B, C, D : 32-bit

Others

- Multi2
- SAFER (Secure And Fast Encryption Routine)-64,128
- Lion & Bear
- TEA (Tiny Encryption Algorithm)
- CAST (Carlisle Adams & Stafford Tavares)
Summary of block ciphers

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Year</th>
<th>Country</th>
<th>Pt/Ct</th>
<th>Key</th>
<th>Round</th>
</tr>
</thead>
<tbody>
<tr>
<td>DES</td>
<td>1977</td>
<td>USA</td>
<td>64</td>
<td>56</td>
<td>16</td>
</tr>
<tr>
<td>FEAL</td>
<td>1987</td>
<td>Japan</td>
<td>64</td>
<td>64</td>
<td>4,8,16,32</td>
</tr>
<tr>
<td>GOST</td>
<td>1989</td>
<td>Russia</td>
<td>64</td>
<td>256</td>
<td>32</td>
</tr>
<tr>
<td>idea</td>
<td>1990</td>
<td>Swiss</td>
<td>64</td>
<td>128</td>
<td>8</td>
</tr>
<tr>
<td>LOKI</td>
<td>1991</td>
<td>Australia</td>
<td>64</td>
<td>64</td>
<td>16</td>
</tr>
<tr>
<td>SKIPJACK</td>
<td>1990</td>
<td>USA</td>
<td>64</td>
<td>80</td>
<td>32</td>
</tr>
<tr>
<td>MISTY</td>
<td>1996</td>
<td>Japan</td>
<td>64</td>
<td>128</td>
<td>&gt;8</td>
</tr>
<tr>
<td>SEED</td>
<td>1998</td>
<td>Korea</td>
<td>128</td>
<td>128</td>
<td>16</td>
</tr>
</tbody>
</table>

AES requirements

- **Block cipher**
  - 128-bit blocks
  - 128/192/256-bit keys
- **Worldwide-royalty free**
- **More secure than Triple DES**
- **More efficient than Triple DES**
AES Calendar

- Jan. 2, 1997: Announcement of intent to develop AES and request for comments
- Sep. 12, 1997: Formal call for candidate algorithms
- Aug. 20-22, 1998: First AES Candidate Conference and beginning of Round 1 evaluation (15 algorithms), Rome, Italy
- Mar. 22-23, 1999: Second AES Candidate Conference, NY, USA
- Sep. 2000: Final AES selection (Rijndael)

AES1 algorithms

15 algorithms are proposed at AES1 conference

<table>
<thead>
<tr>
<th>Cipher</th>
<th>Submitted by</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAS-1-256</td>
<td>Entrust</td>
<td>Canada</td>
</tr>
<tr>
<td>CrypTON</td>
<td>Future Systems</td>
<td>Korea¹</td>
</tr>
<tr>
<td>Deal</td>
<td>Outerbridge</td>
<td>Canada¹</td>
</tr>
<tr>
<td>DFC</td>
<td>ENS-CNRS</td>
<td>France</td>
</tr>
<tr>
<td>E2</td>
<td>NIST¹</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>USA¹</td>
</tr>
<tr>
<td>RC3</td>
<td>RSA</td>
<td>USA¹</td>
</tr>
<tr>
<td>Rijndael*</td>
<td>Daemen, Rijseen</td>
<td>Belgium²</td>
</tr>
<tr>
<td>Safer+</td>
<td>Cylinda</td>
<td>USA¹</td>
</tr>
<tr>
<td>Serpent*</td>
<td>Anderson, Biham, Knudsen</td>
<td>UK, Israel, Norway</td>
</tr>
<tr>
<td>Twofish*</td>
<td>Counterpump</td>
<td>USA¹</td>
</tr>
</tbody>
</table>

* Placed in the public domain  † not foreign designer  ‡ foreign influence

©ICU Kwangjo Kim
AES Round 2 Algorithms

- After AES2 conference, NIST selected the following 5 algorithms as the round 2 candidate algorithm.

<table>
<thead>
<tr>
<th>Algorithm Name</th>
<th>Submitter Name(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MARS</td>
<td>IBM (represented by Nevenko Zunic)</td>
</tr>
<tr>
<td>Rijndael(I)</td>
<td>Joan Daemen, Vincent Rijmen</td>
</tr>
<tr>
<td>Serpent</td>
<td>Ross Anderson, Eli Biham, Lars Knudsen</td>
</tr>
<tr>
<td>Twofish</td>
<td>Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, Niels Ferguson</td>
</tr>
</tbody>
</table>

Rijndael(I)

- Proposed by Joan Daemen, Vincent Rijmen(Belgium)
- Design choices
  - Square type
  - Three distinct invertible uniform transformations(Layers)
    - Linear mixing layer : guarantee high diffusion
    - Non-linear layer : parallel application of S-boxes
    - Key addition layer : XOR the round key to the intermediate state
  - Initial key addition, final key addition
- Representation of state and key
  - Rectangular array of bytes with 4 rows (square type)
  - Nb : number of column of the state
  - Nk : number of column of the cipher key
Rijndael(II)

\[
\begin{array}{cccc}
\sigma_{0,0} & \sigma_{0,1} & \sigma_{0,2} & \sigma_{0,3} \\
\sigma_{1,0} & \sigma_{1,1} & \sigma_{1,2} & \sigma_{1,3} \\
\sigma_{2,0} & \sigma_{2,1} & \sigma_{2,2} & \sigma_{2,3} \\
\sigma_{3,0} & \sigma_{3,1} & \sigma_{3,2} & \sigma_{3,3} \\
\end{array}
\]

State (\(Nb=6\))

\[
\begin{array}{cccc}
\kappa_{0,0} & \kappa_{0,1} & \kappa_{0,2} & \kappa_{0,3} \\
\kappa_{1,0} & \kappa_{1,1} & \kappa_{1,2} & \kappa_{1,3} \\
\kappa_{2,0} & \kappa_{2,1} & \kappa_{2,2} & \kappa_{2,3} \\
\kappa_{3,0} & \kappa_{3,1} & \kappa_{3,2} & \kappa_{3,3} \\
\end{array}
\]

Key (\(Nk=4\))

\[
\begin{array}{|c|c|c|}
\hline
\text{Nr} & \text{Nb} = 4 & \text{Nb} = 6 & \text{Nb} = 8 \\
\hline
\text{Nk} = 4 & 10 & 12 & 14 \\
\text{Nk} = 6 & 12 & 12 & 14 \\
\text{Nk} = 0 & 14 & 14 & 14 \\
\hline
\end{array}
\]

Number of rounds (\(Nr\))

Rijndael(III)

Rijndael(State,CipherKey)
{
  KeyExpansion(CipherKey,ExpandedKey);
  AddRoundKey(State,ExpandedKey);
  For( i=1 ; i<Nr ; i++ ) Round(State,ExpandedKey + Nb*i);
  FinalRound(State,ExpandedKey + Nb*Nr);
}

Round(State,RoundKey)
{
  ByteSub(State);
  ShiftRow(State);
  MixColumn(State);
  AddRoundKey(State,RoundKey);
}

FinalRound(State,RoundKey)
{
  ByteSub(State);
  ShiftRow(State);
  AddRoundKey(State,RoundKey);
}
### Comparison of AES2 algorithms (I)

- **Encryption speed analysis by NIST**

![Graph showing comparison of encryption speeds](image)

### Comparison of AES2 algorithms (II)

- **Java Implementation by A. Sterbenz (Graz Univ.)**

<table>
<thead>
<tr>
<th>Encryption Speed</th>
<th>DES (64 bit)</th>
<th>Triple DES (168 bit)</th>
<th>IDEA</th>
<th>MARS</th>
<th>RC6</th>
<th>Rijndael</th>
<th>Serpent</th>
<th>Twofish</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DES16 bit key</strong></td>
<td>15.08 ms</td>
<td>41.17 ms</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td><strong>DES192 bit key</strong></td>
<td>12.89 ms</td>
<td>19.78 ms</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td><strong>DES256 bit key</strong></td>
<td>12.89 ms</td>
<td>26.21 ms</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td><strong>2DES16 bit key</strong></td>
<td>151.19 ms</td>
<td>41.17 ms</td>
<td>130.18 ms</td>
<td>196.63 ms</td>
<td>2.03 ms</td>
<td>1.86 ms</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td><strong>2DES192 bit key</strong></td>
<td>130.18 ms</td>
<td>26.21 ms</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td><strong>2DES256 bit key</strong></td>
<td>12.89 ms</td>
<td>26.21 ms</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td><strong>3DES16 bit key</strong></td>
<td>151.19 ms</td>
<td>41.17 ms</td>
<td>130.18 ms</td>
<td>196.63 ms</td>
<td>2.03 ms</td>
<td>1.86 ms</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td><strong>3DES192 bit key</strong></td>
<td>130.18 ms</td>
<td>26.21 ms</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td><strong>3DES256 bit key</strong></td>
<td>12.89 ms</td>
<td>26.21 ms</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
</tbody>
</table>

© ICU Kwangjo Kim
Comparison of AES2 algorithms (III)

- Smart Card Implementation by F. Sano (Toshiba)

<table>
<thead>
<tr>
<th>Cipher</th>
<th>RAM (bytes)</th>
<th>ROM (bytes)</th>
<th>Time (clock)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Encrypt</td>
<td>Schedule</td>
<td>Encrypt + Schedule</td>
</tr>
<tr>
<td>MARS</td>
<td>972</td>
<td>5,168</td>
<td>45,588</td>
</tr>
<tr>
<td>RC5</td>
<td>156</td>
<td>1,060</td>
<td>2,34,736</td>
</tr>
<tr>
<td>Rijndael</td>
<td>66</td>
<td>1,980</td>
<td>125,494</td>
</tr>
<tr>
<td>Serpent</td>
<td>164</td>
<td>3,937</td>
<td>71,924</td>
</tr>
<tr>
<td>Twofish</td>
<td>90</td>
<td>2,308</td>
<td>33,877</td>
</tr>
</tbody>
</table>

* : omit to check “weak” in the key schedule

Comparison of AES2 algorithms (IV)

- CMOS ASIC Implementation by Ichikawa (Mitsubishi)

**Table 4.1 Hardware evaluation results**

<table>
<thead>
<tr>
<th>Algorithm name</th>
<th>Enclosure &amp; Description</th>
<th>Key Schedule</th>
<th>Total</th>
<th>Key setup time</th>
<th>Critical path</th>
<th>Throughput [Mbps]</th>
</tr>
</thead>
<tbody>
<tr>
<td>DES</td>
<td>12,394</td>
<td>12,200</td>
<td>54,895</td>
<td>-</td>
<td>55.11</td>
<td>1161.31</td>
</tr>
<tr>
<td>Triple-DES</td>
<td>124,888</td>
<td>22,200</td>
<td>148,147</td>
<td>-</td>
<td>157.07</td>
<td>407.4</td>
</tr>
<tr>
<td>MARS</td>
<td>900,096</td>
<td>2,365,896</td>
<td>2,395,896</td>
<td>1740.99</td>
<td>667.49</td>
<td>223.55</td>
</tr>
<tr>
<td>RC5</td>
<td>111,611</td>
<td>901,096</td>
<td>1,643,097</td>
<td>2122.29</td>
<td>627.57</td>
<td>203.96</td>
</tr>
<tr>
<td>Rijndael</td>
<td>518,506</td>
<td>691,088</td>
<td>692,688</td>
<td>57.30</td>
<td>85.64</td>
<td>1950.03</td>
</tr>
<tr>
<td>Serpent</td>
<td>289,633</td>
<td>200,096</td>
<td>483,730</td>
<td>114.07</td>
<td>137.4</td>
<td>931.58</td>
</tr>
<tr>
<td>Twofish</td>
<td>269,195</td>
<td>231,682</td>
<td>501,877</td>
<td>16.38</td>
<td>32.68</td>
<td>394.08</td>
</tr>
</tbody>
</table>