• Advanced Encryption Standard (AES) Program was launched by the National Institute of Standards and Technology (NIST) in 1997.

• It aims to set the encryption standard of the coming years by finding an algorithm that will replace the aging DES.

• NIST set the following minimal requirements for the proposed algorithms.

  • Block cipher operating on 128-bit long blocks.
  • Support for 128, 192 and 256-bit key lengths.
  • No weak keys.
  • Efficient in software implementation (!)
DES/AES Overview

• 15 candidate algorithms were submitted for the AES effort.

• In August 1999, the following 5 algorithms were selected as finalists.
  
  • MARS by IBM Corporation
  • RC6 by RSA Data Security Inc.
  • Rijndael by Joan Daemen and Vincent Rijmen
  • Serpent by Ross Anderson, Eli Biham and Lars Knudsen
  • Twofish by Counterpane Systems

• This study includes brief explanations of the AES candidates as well as their performance specifications on the new IA-64 platform,
MARS

- Features data dependent rotations, multiplications, S-box substitutions and key-mixing operations in its structure.
- Heterogeneous multi-layer design.
- Rapid spread of interdependency between key and data bits.
RC6

- Simple, compact design featuring data-dependent rotations.
- Use of multiplication for diffusion.
- Requires platforms supporting fast multiplications for efficient implementation.

RC6 has the following simple Pseudo-code.

\[
\begin{align*}
B &= B + S[0]; \\
D &= D + S[1]; \\
\text{for } i = 1 \text{ to } r, \text{ do } \{ \\
&\quad t = (B \times (2B + 1)) \ll \lg w; \\
&\quad u = (D \times (2D + 1)) \ll \lg w; \\
&\quad A = ((A \oplus t) \ll\ll u) + S[2i]; \\
&\quad C = ((C \oplus u) \ll\ll t) + S[2i + 1]; \\
&\quad (A; B; C; D) = (B; C; D; A); \\
&\} \\
A &= A + S[2r + 2]; \\
C &= C + S[2r + 3];
\end{align*}
\]

Fig. 1. Encryption with RC6-w/r/b. Here \( f(x) = x \times (2x + 1) \).
• Features Column Mixing, Byte Substitution and Data Shifting operations in its structure (see figures).
• Has very efficient table-lookup implementation.
• Low memory requirements.

Rijndael has the following Pseudo-code:

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

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);
}
```
The Serpent Cipher:

\[
\begin{align*}
B[0] &= \text{IP}(P); \\
B[i+1] &= R_i(B[i]); \\
C &= \text{FP}(B[32]);
\end{align*}
\]

Where,

\[
\begin{align*}
R_i(X) &= L(S_i(X+K[i])); & i &= 0,1,\ldots,30 \\
R_i(X) &= S_i(X+K[i]) \oplus K[32]; & i &= 31
\end{align*}
\]

- Diffusion is achieved by the Linear Transformation steps.
- Confusion achieved by S-box substitutions and XOR operations
- Input and output whitening achieved by the initial and final XOR operations
- Ultra-conservative in its security margin.
- Main security is provided by the Nonlinear substitution phase (S-box substitution).
- The algorithm, as it is, is optimized for parallel implementation on 32-bit processors.
- Fast performance across most platforms.

- Features key-dependent S-boxes.

- Can be optimized for speed, key setup, memory, code size in software, or space in hardware.

The boxes labeled MDS represent matrix multiplication with the following matrix:

\[
\begin{pmatrix}
01 & \text{EF} & 5B & 5B \\
5B & \text{EF} & \text{CF} & 01 \\
\text{EF} & 5B & 01 & \text{EF} \\
\text{EF} & 01 & \text{EF} & 5B
\end{pmatrix}
\]
The figure above is for a comparison of the performances of the five candidates on the IA-64 platform. For each algorithm we report:

- The number of instructions to execute for encryption of one block of data
- The number of clock cycles required for the encryption of one block of data
• The security level provided by all of the algorithms are the same. The brute-force attacks on all of these algorithms require $2^{128}$ attempts.

• Software performance and security of the algorithms are only two measures that will be judged by the NIST in the selection of the winner algorithm(s). Some other important performance metrics to be considered are:
  
  • cost
  • flexibility
  • simplicity.

• The algorithms also need to be suitable for applications such as use in stream ciphers and hash functions and must also be small and compact enough to allow implementation in smart cards.