Advanced Encryption Standard

AES is a block cipher that comes from the cipher Rijndael invented by Joan Daemen and Vincent Rijmen. It won the AES contest in 2000.

It is a block cipher. The block length and key length for Rijndael can be chosen independently to be 128, 192 or 256 bits. AES specifies that the block size is 128 bits.

It has 10, 12 or 14 rounds, depending on the key length. The rounds do not have a Feistel structure.

It was designed to be simple, to be resistant against all known attacks and to have fast and compact code on many platforms.

Rijndael was based on an older cipher called Square. The was one known attack on Square, and the designers of Rijndael fixed it in Rijndael.

The design specifications of Rijndael are public.

In 2009, the first known attack on AES was published. A related-key attack takes 2^{119} steps to break 256-bit AES and 2^{176} steps to break 192-bit AES. This attack does not work on 128-bit AES, whose fastest known attack is brute force and takes 2^{128} steps.

None of these attacks is feasible.

There are attacks on AES with a reduced number of rounds.

Rijndael has 10, 12 or 14 rounds, if the key length is 128, 192 or 256 bits.

Different parts of the Rijndael algorithm operate on the intermediate result, called the *State*. The State is a square array of bytes with four rows and four columns.

The Key is expanded and placed in an array `W[4*(Nr+1)]`, where Nr is the number of rounds. Let Nk be the number of 32-bit words in the key (4, 6, or 8). The first Nk words of `W` are the Key. Each subsequent word is the XOR of the previous word and the word `Nk` words back in the array, except that words whose subscript is a multiple of `Nk` have the previous word transformed before the XOR.

Each round of Rijndael consists of four different transformations:

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

The FinalRound omits the MixColumn.

The complete Rijndael cipher consists of:

```
Rijndael(State, CipherKey)
{
KeyExpansion(CipherKey, ExpandedKey);
AddRoundKey(State, ExpandedKey);
For (i=1; i<Nr; i++)
       Round(State, ExpandedKey + 4*i);
FinalRound(State, ExpandedKey + 4*Nr);
}
```

`ByteSub(State)` transforms each byte in the State in a non-linear way using a look-up table (like an S-box in DES).

`ShiftRow(State)` is a circular left shift of each row in the State by various byte offsets. Mixed up the bytes in a row.

In `MixColumn(State)` the four bytes of each column of the State are transformed using an invertible arithmetic operation involving polynomials with coefficients in a finite field. Mixed up the bytes in a column.

`AddRoundKey(State, RoundKey)` is simply an XOR of State with RoundKey. This operation is also done once before the first round begins.