**ADVANCED ENCRYPTION STANDARD**

**The Origins of AES**

The principal drawback of 3DES (which was recommended in 1999, Federal Information Processing Standard FIPS PUB 46-3 as new standard with 168-bit key) is that the algorithm is relatively sluggish in software. A secondary drawback is the use of 64-bit block size. For reasons of both efficiency and security, a larger block size is desirable.

In 1997, National Institute of Standards and Technology NIST issued a call for proposals for a new Advanced Encryption Standard (AES), which should have security strength equal to or better than 3DES, and significantly improved efficiency. In addition, NIST also specified that AES must be a symmetric block cipher with a block length of 128 bits and support for key lengths of 128, 192, and 256 bits.

In a first round of evaluation, 15 proposed algorithms were accepted. A 2nd round narrowed to 5 algorithms. NIST completed its evaluation process and published a final standard (FIPS PUB 197) in November, 2001. NIST selected Rijndael as the proposed AES algorithm. The 2 researches of AES are Dr. Joan Daemon and Dr. Vincent Rijmen from Belgium.

AES Evaluation

**Security** – 128 minimal key size provides enough security

**Cost** – AES should have high computational efficiency

**The Origins of AES (Cont 1)**



Algorithm and implementation characteristics – this includes variety of considerations, including flexibility, suitability for hardware and software implementations, simplicity

**The Origins of AES (Cont 2)**



Additional criteria include: general security, software implementations, restricted-space environments, hardware implementations, attacks on implementation (timing attacks), encryption versus decryption, key agility, flexibility, potential for instruction-level parallelism.

**The Origins of AES (Cont 3)**



**The Origins of AES (Cont 4)**



**THE AES CIPHER**



A number of AES parameters depend on the key length (Table 5.3). In the description of this section, we assume the key length of 128 bits.

Figure 5.1 shows the overall structure of AES.

**OVERALL STRUCTURE OF AES**





The input to the encryption and decryption algorithm is a single 128-bit block, this block, in FIPS PUB 197, is depicted as a square matrix of bytes. This block is copied into the State array, which is modified at each stage of encryption or decryption. After the final stage, State is copied to an output matrix. These operations are depicted in Figure 5.2a:

**OVERALL STRUCTURE OF AES (Cont 1)**



Similarly, the 128-bit is depicted as a square matrix of bytes. This key is expanded into an array of key schedule words; each word is 4 bytes and the total key schedule is 44 words for the 128-bit key (Figure 5.2b). Ordering of bytes within a matrix is by column.

Before delving into details, we can make several comments about overall AES structure:

1. This cipher is not a Feistel structure.
2. The key that is provided as input is expanded into an array of 44 words (32-bits each), w[i]. 4 distinct words (128 bits) serve as a round key for each round; these are indicated in Fig. 5.1
3. 4 different stages are used, 1 permutation and 3 of substitution:
* Substitute bytes – Uses an S-box to perform a byte-to-byte substitution of the block
* Shift rows – A simple permutation
* Mix columns – A substitution that makes use of arithmetic over GF(28).
* Add round key – A simple bitwise XOR of the current block with the portion of the expanded key
1. The structure is quite simple. Figure 5.3 depicts the structure of a full encryption round

**OVERALL STRUCTURE OF AES (Cont 2)**



1. Only the Add Round Key stage uses the key. Any other stage is reversible without knowledge of the key.
2. The Add Round Key is a form of Vernam cipher and by itself would not be formidable. The other 3 stages together provide confusion, diffusion, and nonlinearity, but by themselves would provide no security because they do not use the key. We can view the cipher as alternating operations of XOR encryption (Add Round Key), followed by scrambling of the block.
3. Each stage is easily reversible
4. Decryption uses the same keys but in the reverse order. Decryption is not identical to encryption
5. At each horizontal point (e.g., the dashed line) in Figure 5.1, State is the same for both encryption and decryption
6. The final round of both encryption and decryption consists of only 3 stages; it is the consequence of the particular structure of AES.

**OVERALL STRUCTURE OF AES (Cont 3)**

As was mentioned in Chapter 4, AES uses arithmetic in the finite field GF(28), with the irreducible polynomial .

**Substitute Byte Transformation. Forward and Inverse Transformation**

The Forward substitute byte transformation, called SubBytes, is a simple table lookup (Figure 5.4a).



**Substitute Byte Transformation. Forward and Inverse Transformation (Cont 2)**

AES defines a 16x16 matrix of byte values, called an S-box (Table 5.4a), that contains a permutation of all possible 256 8-bit values. Each individual byte of State is mapped into a new byte in the following way: The leftmost 4 bits are used as a row value and the rightmost 4 bits are used as a column value. These row and column values serve as indexes into the S-box to select a unique 8-bit output value. For example, the hexadecimal value {95} references row 9, column 5 of the S-box, which contains the value {2a}:

**Substitute Byte Transformation. Forward and Inverse Transformation (Cont 3)**



The S-box is constructed in the following fashion:

1. Initialize the S-box with the byte values in ascending order row by row. Thus, the value of the byte at row x, column y is {xy}

**Substitute Byte Transformation. Forward and Inverse Transformation (Cont 4)**

1. Map each byte in the S-box to its multiplicative inverse in the finite field GF(28); the value {00} is mapped to itself.
2. Consider that each byte in the S-box consists of 8 bits labeled (b7,b6,b5,b4,b3,b2,b1,b0). Apply the following transformation to each bit of each byte in the S-box:

 (5.1)

where ci is the i-th bit of byte c with the value {63}, that is, (c7c7c5c4c3c2c1c0)=(01100011). The prime  indicates that the variable is to be updated by the value on the right. The AES standard depicts this transformation in matrix form as follows:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| B0’ |  | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |  | B0 |  | 1 |  |
| B1’ |  | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |  | B1 |  | 1 |  |
| B2’ |  | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |  | B2 |  | 0 |  |
| B3’ | = | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | x | B3 | + | 0 | (5.2) |
| B4’ |  | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |  | B4 |  | 0 |  |
| B5’ |  | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |  | B5 |  | 1 |  |
| B6’ |  | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 |  | B6 |  | 1 |  |
| B7’ |  | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |  | B7 |  | 0 |  |

Each element in the product matrix is the bitwise XOR of elements of one row and one column. Further, the final addition, shown in (5.2), is a bitwise XOR.

As an example, consider the input value {95}. The multiplicative inverse in GF(28) is {95}-1 ={8a}, which is 10001010 in binary. Using equation (5.2),

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |  | 0 |  | 1 |  | 1 |  | 1 |  | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |  | 1 |  | 1 |  | 0 |  | 1 |  | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |  | 0 |  | 0 |  | 0 |  | 0 |  | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | x | 1 | + | 0 | = | 1 | + | 0 | = | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |  | 0 |  | 0 |  | 0 |  | 0 |  | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |  | 0 |  | 1 |  | 0 |  | 1 |  | 1 |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 |  | 0 |  | 1 |  | 1 |  | 1 |  | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |  | 1 |  | 0 |  | 0 |  | 0 |  | 0 |

The result is {2a}, which should appear in row {09} column {05} of the S-box. This is verified by checking Table 5.4a.