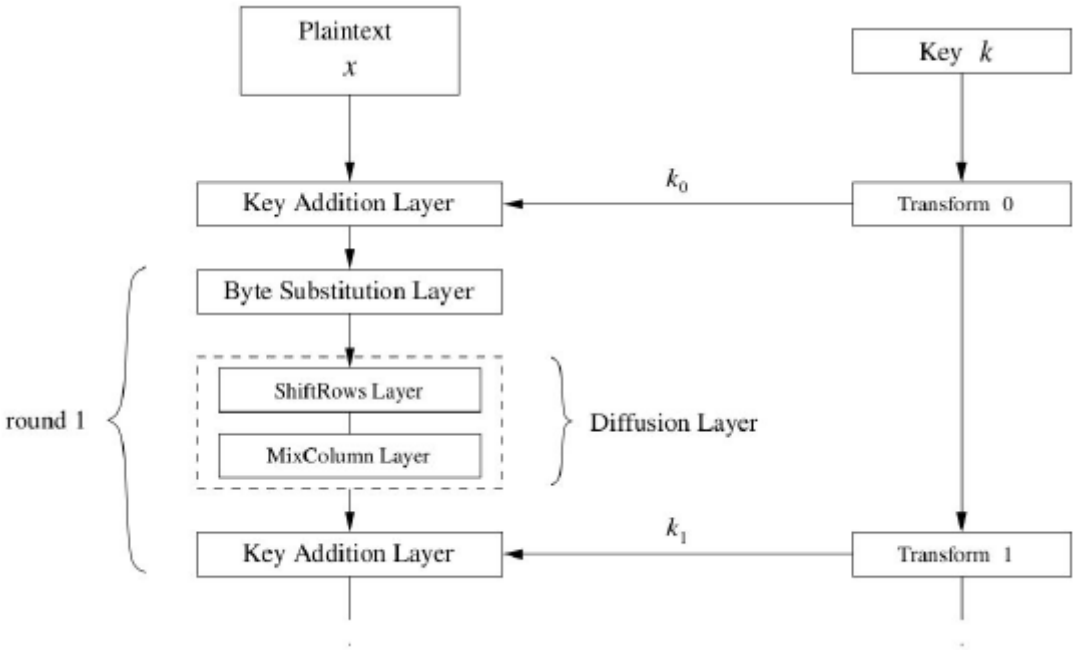
3. Symmetric ciphers II (AES)

3.1 Standard

Standard for NIST; key: 128/192/256 bytes

input: block of 128 bytes.

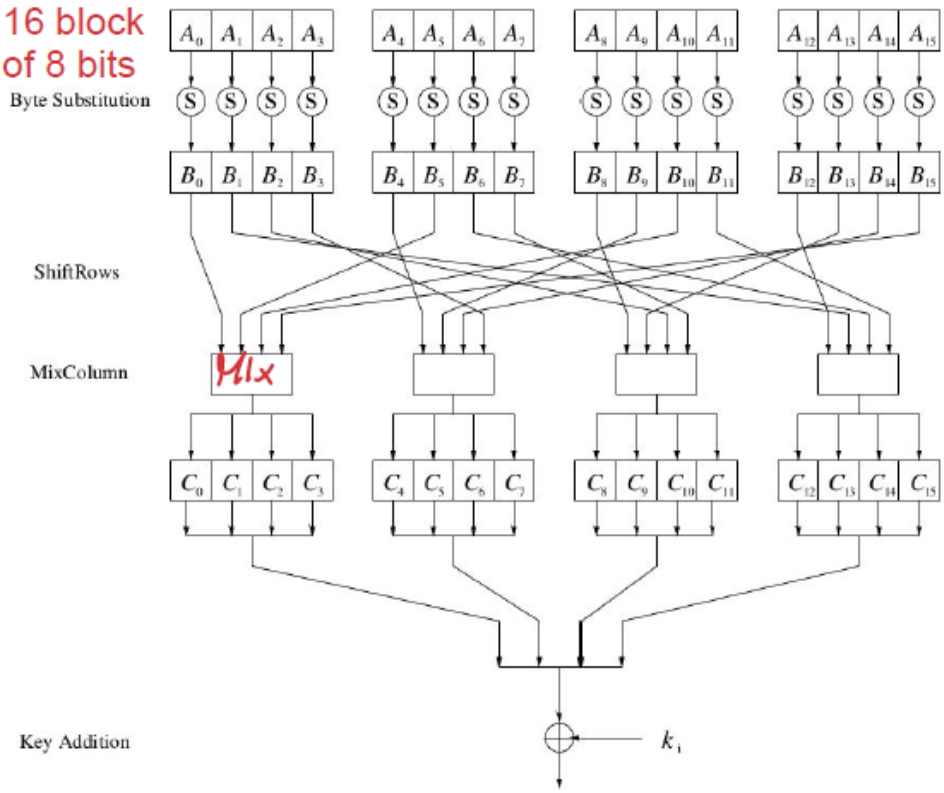


10/12/14 rounds

3.2 Layers

In the last round we skip MixColumn Layer

1. ByteSub → CONFUSION
2. ShiftRow → DIFFUSION
3. MixColumn → DIFFUSION
4. Key Addition → KEY WHITENING



3.3 Internal structure

- **Byte-oriented** cipher.
- Based on a **substitution-permutation network**
- A (A0,.....,A15, 16-byte (128bit) input) can be draw as a 4×4 matrix

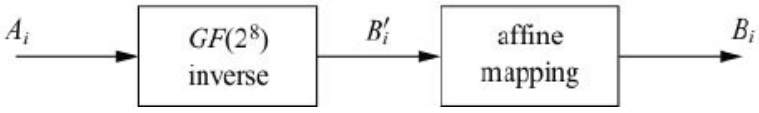
3.3.1 Byte substitution Layer

Independent for every byte.

16 S-Boxes identical, non linear and bijective (S-Box can be uniquely reversed)

Confusion: 1 bit in Ai can affect 3/4 bits in Bi

The S-Box perform two operations:



Ai = 1100 0010 ⇒ Ai(x) = x^7 + x^6 + x

The first step computes the inverse (which provides the non linearity in AES):

$$B'_i(x) = A(x)^{-1} \qquad P(x) = x^8 + x^4 + x^3 + x + 1$$

such that: $B'_i(x) \cdot A(x)^{-1} \equiv 1 \bmod P(x)$

AES irreducible polynomial

The second step computed in the S-Box is an affine mapping (this is done to destroy some algebraic properties that could be exploited by an attacker):

$$\begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{pmatrix}_{B_i(x)} \equiv \begin{pmatrix} 1&0&0&0&1&1&1&1 \\ 1&1&0&0&0&1&1&1 \\ 1&1&1&0&0&0&1&1 \\ 1&1&1&1&0&0&0&1 \\ 1&1&1&1&1&0&0&0 \\ 0&1&1&1&1&1&0&0 \\ 0&0&1&1&1&1&1&0 \\ 0&0&0&1&1&1&1&1 \end{pmatrix} \begin{pmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{pmatrix}_{B'_i(x)} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} \bmod 2.$$

```
#include <stdint.h>

#define ROTL8(x,shift) ((uint8_t) ((x) << (shift)) |
#define ((x) >> (8 - (shift))))

void initialize_aes_sbox(uint8_t sbox[256]) {
  uint8_t p = 1, q = 1;

  /* loop invariant: p * q == 1 in the Galois field */
  do {
    /* multiply p by 3 */
    p = p ^ (p << 1) ^ (p & 0x80 ? 0x1B : 0);

    /* divide q by 3 (equals multiplication by 0xf6) */
    q ^= q << 1;
    q ^= q << 2;
    q ^= q << 4;
    q ^= q & 0x80 ? 0x09 : 0;

    /* compute the affine transformation */
    uint8_t xformed = q ^ ROTL8(q, 1) ^ ROTL8(q, 2) ^ ROTL8(q, 3)
                      ^ ROTL8(q, 4);

    sbox[p] = xformed ^ 0x63;
  } while (p != 1);

  /* 0 is a special case since it has no inverse */
  sbox[0] = 0x63;
}
```

3.3.2 Diffusion Layer

Diffusion: given a byte with some bit flips, it will spread the effect on 32 bits from the state.

ShiftRows Sublayer: Permutation of the data on a byte level (shift bit on the left cyclically on every row)
MixColumn Sublayer: Matrix operation which combines ("mixes") blocks of four bytes performs a linear operation on state matrices A, B, i.e.,

$$\begin{pmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \cdot \begin{pmatrix} B_0 \\ B_5 \\ B_{10} \\ B_{15} \end{pmatrix}$$

C0 = 02*B0 + 03*B5 + 01*B10 + 01*B15

C0 = x*B0 + (x+1)*B5 + 1*B10 + 1*B15

E.g., B=(25, ..., 25)

$$\begin{aligned} 02 \cdot 25 &= x \cdot (x^5 + x^2 + 1) \\ &= x^6 + x^3 + x, \\ 03 \cdot 25 &= (x+1) \cdot (x^5 + x^2 + 1) \\ &= (x^6 + x^3 + x) + (x^5 + x^2 + 1) \\ &= x^6 + x^5 + x^3 + x^2 + x + 1. \end{aligned}$$

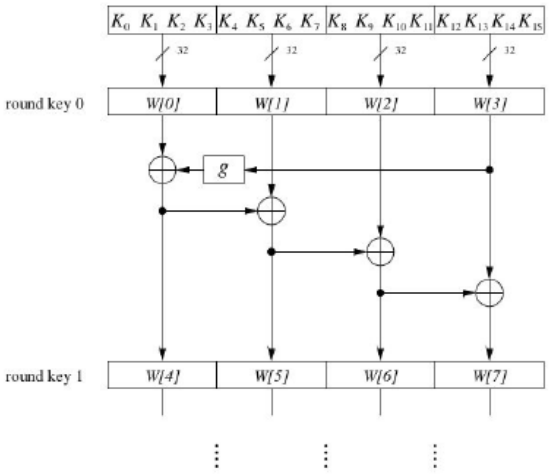
$$\begin{array}{rl} 01 \cdot 25 = & x^5 + x^2 + 1 \\ 01 \cdot 25 = & x^5 + x^2 + 1 \\ 02 \cdot 25 = & x^6 + x^3 + x \\ 03 \cdot 25 = & x^6 + x^5 + x^3 + x^2 + x + 1 \\ \hline C_i = & x^5 + x^2 + 1, \end{array}$$

3.3.3 Key Addition Layer

- 16-byte state matrix C and 16-byte subkey ki
  - Output: C ⊕ ki
  - The subkeys are generated in the key schedule recursively from the original

Each round has 1 subkey, plus 1 subkey at the beginning of AES

Key whitening: Subkey is used both at the input and output of AES



- Word-oriented: 1 word = 32 bits
- 11 subkeys are stored in W[0]...W[3], W[4]...W[7], ... , W[40]...W[43]
- First subkey W[0]...W[3] is the original AES key

For 128 bits

Function g rotates its four input bytes and performs a bytewise S-Box substitution ⇒ nonlinearity

RC (Round coefficient) is only added to the leftmost byte and varies from round to round (equals the number of the round in binary)

3.4 Decryption

All layers must be inverted for decryption.