C | SecureAES

- High-Performance AES Encryption in C -

Ji Yong-Hyeon

Department of Information Security, Cryptology, and Mathematics
College of Science and Technology
Kookmin University

December 16, 2023

Contents

Chapter 1

Block Cipher AES-128

1.1 Overview of Advanced Encryption Standard

- AddRoundKey : $\{0,1\}^{128} \times \{0,1\}^{128} \rightarrow \{0,1\}^{128}$
- SubBytes : $\{0,1\}^{128} \rightarrow \{0,1\}^{128}$
- ShiftRows : $\{0,1\}^{128} \rightarrow \{0,1\}^{128}$
- MixColumns : $\{0,1\}^{128} \rightarrow \{0,1\}^{128}$

Algorithm 1: Encryption of AES-128

Input: block src $\in \{0,1\}^{128}$, round-keys $\{rk_i\}_{i=0}^{11}$ ($rk_i \in \{0,1\}^{128}$)
Output: block dst $\in \{0,1\}^{128}$

```
1  t ← src;
2  t ← AddRoundKey(t, rk_0);
3  for i ← 1 to 9 do
4  |   t ← SubBytes(t);
5  |   t ← ShiftRows(t);
6  |   t ← MixColumns(t);
7  |   t ← AddRoundKey(t, rk_i);
8  end
9  t ← SubBytes(t);
10 t ← ShiftRows(t);
11 t ← AddRoundKey(t, rk_10);
12 dst ← t;
13 return dst;
```

1.2 Functions and Constants used in AES

1.2.1 Key Expansion

- RotWord : $\{0,1\}^{32} \to \{0,1\}^{32}$ is defined by

$$\text{RotWord}\left(X_0 \parallel X_1 \parallel X_2 \parallel X_3\right) := X_1 \parallel X_2 \parallel X_3 \parallel X_0 \quad \text{for} \quad X_i \in \{0,1\}^8 .$$

Code 1.1: RotWord rotates the input word left by one byte

```
u32 RotWord(u32 word) {
    return (word << 0x08) | (word >> 0x18);
}
```

- SubWord : $\{0,1\}^{32} \to \{0,1\}^{32}$ is defined by

$$\text{SubWord}(X_0 \parallel X_1 \parallel X_2 \parallel X_3) := s(X_0) \parallel s(X_1) \parallel s(X_2) \parallel s(X_3) \quad \text{for} \quad X_i \in \{0,1\}^8 .$$

Here, $s : \{0,1\}^8 \to \{0,1\}^8$ is the S-box.

Code 1.2: SubWord applies the S-box to each byte of the input word

```
u32 SubWord(u32 word) {
    return (u32)s_box[word >> 0x18] << 0x18 |
        (u32)s_box[(word >> 0x10) & 0xFF] << 0x10 |
        (u32)s_box[(word >> 0x08) & 0xFF] << 0x08 |
        (u32)s_box[word & 0xFF];
}
```

- Round Constant Rcon:

The constant $\texttt{Rcon}_i \in \mathbb{F}_{2^8}$ used in generating the i-th round key corresponds to the value of x^{i-1} in the binary finite field $\mathbb{F}_{2^8}$ and is as follows:

i	1	2	3	4	5	6	7	8	9	10
$\texttt{Rcon}_i$	0x01	0x02	0x04	0x08	0x10	0x20	0x40	0x80	0x1b	0x36

Code 1.3: Rcon Array Declaration

```
static const u32 Rcon[10] = {
    0x01000000, 0x02000000, 0x04000000, 0x08000000,
    0x10000000, 0x20000000, 0x40000000, 0x80000000,
    0x1b000000, 0x36000000
};
```

Algorithm 2: Key Schedule (AES-128)

Input: key $k = (k_0, \ldots, k_{15})$ $(k_i \in \{0,1\}^8)$; // $k \in \{0,1\}^{128}$ is 16-byte

Output: round-keys $\{rk_i\}_{i=0}^{43}$ $(rk_i \in \{0,1\}^{32})$; // $\{rk_i\}_{i=0}^{43} \in \{0,1\}^{1408}$ is 176-byte

$rk_0 \leftarrow k_0 \parallel k_1 \parallel k_2 \parallel k_3$;
$rk_1 \leftarrow k_4 \parallel k_5 \parallel k_6 \parallel k_7$;
$rk_2 \leftarrow k_8 \parallel k_9 \parallel k_{10} \parallel k_{11}$;
$rk_3 \leftarrow k_{12} \parallel k_{13} \parallel k_{14} \parallel k_{15}$;
for $i = 4$ **to** 43 **do**
  $t \leftarrow rk_{i-1}$;
  if $i \bmod 4 = 0$ **then**
    /* SubWord $\circ$ RotWord : $\{0,1\}^{32} \rightarrow \{0,1\}^{32}$ */
    $t \leftarrow \text{RotWord}(t)$;
    $t \leftarrow \text{SubWord}(t)$;
    $t \leftarrow t \oplus (\text{Rcon}_{i/4} \parallel \texttt{0x00} \parallel \texttt{0x00} \parallel \texttt{0x00})$;
  end
  $rk_i \leftarrow rk_{i-4} \oplus t$;
end

Code 1.4: AES Key Expansion

```
void KeyExpansion(const u8* key, u32* rKey) {
    u32 temp;
    int i = 0;

    // Copy the input key to the first round key
    while (i < 4) {
        rKey[i] = (u32)key[4*i] << 0x18 |
        (u32)key[4*i+1] << 0x10 |
        (u32)key[4*i+2] << 0x08 |
        (u32)key[4*i+3];
        i++;
    }

    i = 4;

    // Generate the remaining round keys
    while (i < 44) {
        temp = rKey[i-1];
        if (i % 4 == 0) {
            temp = SubWord(RotWord(temp)) ^ Rcon[i/4-1];
        }
        rKey[i] = rKey[i-4] ^ temp;
        i++;
    }
}
```

1.3 Code Structure

1. Rcon Array Declaration
2. Function Definition
3. Variable Declarations and Initial Checks
4. Key Expansion Logic

1.4 Detailed Analysis

1.4.1 Rcon Array Declaration

1.4.2 Function Definition

```
int AES_set_encrypt_key(const unsigned char *userKey, const int bits,
```

1.4.3 Variable Declarations and Initial Checks

```
u32 *rk;
int i = 0;
u32 temp;
if (!userKey || !key)
return -1;
if (bits != 128 && bits != 192 && bits != 256)
return -2;
```

1.4.4 Key Expansion Logic

1. Initial Key Setup
2. Key Expansion based on key size

1.4.5 AddRoundKey

1.4.6 SubBytes

1.4.7 ShiftRows

1.4.8 MiColums

Appendix A

Additional Data A

A.1 Substitution-BOX

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
10	ca	82	...	...	...	...	...	...	...	...	...	...	...	...	...	...
30	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
40	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
50	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
60	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
70	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
80	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
90	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
a0	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
b0	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
c0	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
d0	...	...	...	...	...	...	...	...	...	...	...	...	...	c1	...	...
e0	...	...	...	...	...	...	...	...	...	...	...	...	...	...	28	...
f0	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	16

```
static const u8 s_box[256] = {
    0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5, 0x30,
    0x01, 0x67, 0x2b, 0xfe, 0xd7, 0xab, 0x76, 0xca, 0x82,
    0xc9, 0x7d, 0xfa, 0x59, 0x47, 0xf0, 0xad, 0xd4, 0xa2,
    0xaf, 0x9c, 0xa4, 0x72, 0xc0, 0xb7, 0xfd, 0x93, 0x26,
    0x36, 0x3f, 0xf7, 0xcc, 0x34, 0xa5, 0xe5, 0xf1, 0x71,
    0xd8, 0x31, 0x15, 0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96,
    0x05, 0x9a, 0x07, 0x12, 0x80, 0xe2, 0xeb, 0x27, 0xb2,
    0x75, 0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0,
    0x52, 0x3b, 0xd6, 0xb3, 0x29, 0xe3, 0x2f, 0x84, 0x53,
    0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b, 0x6a, 0xcb,
    0xbe, 0x39, 0x4a, 0x4c, 0x58, 0xcf, 0xd0, 0xef, 0xaa,
    0xfb, 0x43, 0x4d, 0x33, 0x85, 0x45, 0xf9, 0x02, 0x7f,
    0x50, 0x3c, 0x9f, 0xa8, 0x51, 0xa3, 0x40, 0x8f, 0x92,
    0x9d, 0x38, 0xf5, 0xbc, 0xb6, 0xda, 0x21, 0x10, 0xff,
    0xf3, 0xd2, 0xcd, 0x0c, 0x13, 0xec, 0x5f, 0x97, 0x44,
    0x17, 0xc4, 0xa7, 0x7e, 0x3d, 0x64, 0x5d, 0x19, 0x73,
    0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88, 0x46,
    0xee, 0xb8, 0x14, 0xde, 0x5e, 0x0b, 0xdb, 0xe0, 0x32,
    0x3a, 0x0a, 0x49, 0x06, 0x24, 0x5c, 0xc2, 0xd3, 0xac,
    0x62, 0x91, 0x95, 0xe4, 0x79, 0xe7, 0xc8, 0x37, 0x6d,
    0x8d, 0xd5, 0x4e, 0xa9, 0x6c, 0x56, 0xf4, 0xea, 0x65,
    0x7a, 0xae, 0x08, 0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6,
    0xb4, 0xc6, 0xe8, 0xdd, 0x74, 0x1f, 0x4b, 0xbd, 0x8b,
    0x8a, 0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e,
    0x61, 0x35, 0x57, 0xb9, 0x86, 0xc1, 0x1d, 0x9e, 0xe1,
    0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e, 0x94, 0x9b, 0x1e,
    0x87, 0xe9, 0xce, 0x55, 0x28, 0xdf, 0x8c, 0xa1, 0x89,
    0x0d, 0xbf, 0xe6, 0x42, 0x68, 0x41, 0x99, 0x2d, 0x0f,
    0xb0, 0x54, 0xbb, 0x16
};
```