

2019 암호경진대회

2번 문제 : 암호구현 및 최적화

사물인터넷 플랫폼 상에서의 부채널에 강인한 경량 암호 구현을 위해 최근에는 Substitution Permutation Network (SPN) 연산으로 구성된 블록 암호화가 제안되고 있다. 본 문제에서는 새로운 SPN 기반 블록암호화를 8-비트 사물인터넷 플랫폼 상에서 최적화하여 구현해 보도록 한다. 새로운 SPN 기반 블록암호화는 16-비트 평문과 16-비트 비밀키를 가진다. 해당 블록암호의 레퍼런스 코드는 다음과 같다.

```
#include <stdio.h>
#include <stdlib.h>

#define ROUND_NUM 100
typedef unsigned char u8;

u8 key[2] = { 0x12, 0x34 };
u8 rnd[ROUND_NUM * 2] = { 0, };
u8 text[2] = { 0x56, 0x78 };

u8 sbbox[256]={
    0xFF,0xFE,0xFD,0xFC,0xFB,0xFA,0xF9,0xF8,0xF7,0xF6,0xF5,0xF4,0xF3,0xF2,0xF1,0xF0,
    0xEF,0xEE,0xED,0xEC,0xEB,0xEA,0xE9,0xE8,0xE7,0xE6,0xE5,0xE4,0xE3,0xE2,0xE1,0xE0,
    0xDF,0xDE,0xDD,0xDC,0xDB,0xDA,0xD9,0xD8,0xD7,0xD6,0xD5,0xD4,0xD3,0xD2,0xD1,0xD0,
    0xCF,0xCE,0xCD,0xCC,0xCB,0xCA,0xC9,0xC8,0xC7,0xC6,0xC5,0xC4,0xC3,0xC2,0xC1,0xC0,
    0xBF,0xBE,0xBD,0xBC,0xBB,0xBA,0xB9,0xB8,0xB7,0xB6,0xB5,0xB4,0xB3,0xB2,0xB1,0xB0,
    0xAF,0xAE,0xAD,0xAC,0xAB,0xAA,0xA9,0xA8,0xA7,0xA6,0xA5,0xA4,0xA3,0xA2,0xA1,0xA0,
    0x9F,0x9E,0x9D,0x9C,0x9B,0x9A,0x99,0x98,0x97,0x96,0x95,0x94,0x93,0x92,0x91,0x90,
    0x8F,0x8E,0x8D,0x8C,0x8B,0x8A,0x89,0x88,0x87,0x86,0x85,0x84,0x83,0x82,0x81,0x80,
    0x7F,0x7E,0x7D,0x7C,0x7B,0x7A,0x79,0x78,0x77,0x76,0x75,0x74,0x73,0x72,0x71,0x70,
    0x6F,0x6E,0x6D,0x6C,0x6B,0x6A,0x69,0x68,0x67,0x66,0x65,0x64,0x63,0x62,0x61,0x60,
    0x5F,0x5E,0x5D,0x5C,0x5B,0x5A,0x59,0x58,0x57,0x56,0x55,0x54,0x53,0x52,0x51,0x50,
    0x4F,0x4E,0x4D,0x4C,0x4B,0x4A,0x49,0x48,0x47,0x46,0x45,0x44,0x43,0x42,0x41,0x40,
    0x3F,0x3E,0x3D,0x3C,0x3B,0x3A,0x39,0x38,0x37,0x36,0x35,0x34,0x33,0x32,0x31,0x30,
    0x2F,0x2E,0x2D,0x2C,0x2B,0x2A,0x29,0x28,0x27,0x26,0x25,0x24,0x23,0x22,0x21,0x20,
    0x1F,0x1E,0x1D,0x1C,0x1B,0x1A,0x19,0x18,0x17,0x16,0x15,0x14,0x13,0x12,0x11,0x10,
    0x0F,0x0E,0x0D,0x0C,0x0B,0x0A,0x09,0x08,0x07,0x06,0x05,0x04,0x03,0x02,0x01,0x00
};
```

```
void key_gen(u8* rnd, u8* key) {
    u8 key1 = key[0];
    u8 key2 = key[1];
    u8 tmp1, tmp2;

    int i;
    for (i = 0; i < ROUND_NUM; i++) {
        key1 = sbbox[key1];
        key2 = sbbox[key2];

        tmp1 = (key1 | key2) + sbbox[i];
        tmp2 = (key1 & key2) - sbbox[i];

        if(i%2 == 0){
            key1 = tmp1;
            key2 = tmp2;
        }else{
            key1 = tmp2;
            key2 = tmp1;
        }

        rnd[i * 2 + 0] = key1;
        rnd[i * 2 + 1] = key2;
    }
}
```

```
void enc(u8* text, u8* rnd) {
    u8 text1 = text[0];
    u8 text2 = text[1];
    u8 tmp;
    u8 tmp1, tmp2;

    int i;
    for (i = 0; i < ROUND_NUM; i++) {
        text1 = sbbox[text1];
        text2 = sbbox[text2];
```

```

        if(i%2 == 0){
            tmp = text1;
            text1 = text2;
            text2 = tmp;
        }else{
            tmp = text2;
            text2 = text1;
            text1 = tmp;
        }

        tmp1 = ((text1 + text2) ^ text2) & rnd[i * 2 + 0];
        tmp2 = (text1 & (text2 + text1)) ^ rnd[i * 2 + 1];

        text1 = tmp1;
        text2 = tmp2;

    }
    text[0] = text1;
    text[1] = text2;
}

int main(void) {
    key_gen(rnd, key);
    enc(text, rnd);

    return 0;
}

```

해당 블록암호화의 테스트 벡터는 다음과 같으며 해당 결과값이 제대로 나오는지 코드에서 확인하는 부분이 있어야 한다. 아래 값은 16진수로 나타나 있다.

key[0]	key[1]	plain_text[0]	plain_text[1]	cipher_text[0]	cipher_text[1]
0x12	0x34	0x56	0x78	0xC0	0x82

구현 타겟 플랫폼

아두이노 우노 (8-비트 AVR 프로세서) 상에서 구현하며 상세한 정보는 아래와 같다.

<https://store.arduino.cc/usa/arduino-uno-rev3>

프로그래밍의 함수는 다음과 같이 정의하도록 한다.

```
void key_gen(u8* rnd, u8* key){  
    //코드구현  
}  
void enc(u8* text, u8* rnd){  
    //코드구현  
}
```

연산 속도 체크 (벤치마크)는 다음 코드를 활용하도록 한다.

```
u32 time1;  
u32 time2;  
  
time1 = millis();  
for(int i=0;i<10000;i++){  
    key_gen(rnd,key);  
    enc(text,rnd);  
}  
time2 = millis();  
Serial.println((time2-time1));
```

결과물은 다음 2종을 포함한다.

- 아두이노 코드 (테스트 벡터 확인 과정, 벤치마크 과정)
- 문서 (구현 기법 상세, 벤치마크 결과, 테스트 벡터 결과)

평가방법은 다음과 같다.

테스트 벡터 통과 (30점) : 절대 평가

문서화 (30점) : 절대 평가

벤치마크 결과 (40점) : 상대 평가