2018 암호분석경진대회 : 5번 문제 답안

벤치마크 결과 249ms로 ROL 함수를 대신하여 Table로 구현하고, for문을 대신하여 직접 코드에 상수 값으로 대입하는 방법을 사용하여 구현하였습니다.

1. 구현

새로운 ARX기반 블록암호화를 아두이노에서 구현하기 위해 보드 모델은 UNO R3를 사용하였으며 아두이노 1.8.5에서 구현하였습니다.

첫 번째로 **주어진 소스코드와 동일하게 아두이노 코드로 구현**하였습니다. 주어진 코드는 C언어 기반의 코드이므로 약간의 수정을 거쳐 아두이노 코드로 변경하였습니다. 코드는 문제와 거의 동일하기 때문에 자세한 설명없이 프로젝트를 첨부하였습니다.

두 번째로 Table기반 AES를 모방하여 **table로 미리 값을 계산하여 필요한 값에 따라 table의 원소를 저장하는 방법**을 생각하였습니다. key_gen에서 필요한 table의 크기는 key1(8bit)와 key2(8bit)에 출력되는 rnd(20byte)로 총 2^8(key1의 가지 수)*2^8(key2의 가지 수)*20 = 1310720byte의 table 저장 공간이 필요합니다. 하지만 아두이노의 최대 공간을 초과하므로 이 방법을 쓸 수 없습니다. enc 함수에서도 마찬가지로 table을 만들 때 저장공간을 초과합니다.

세 번째로 **ROL 함수를 대신하여 Table로 구현하고 for문을 없애는 방법**을 사용하였습니다.

```
void key_gen(u8* rnd, u8* key){

  u8 key1= key[0];
  u8 key2= key[1];
  u8 tmp1,tmp2;

  key1=key1table[key1];
  key2=key2table[key2];
  tmp1=key1 + key2;
  tmp2=key1 ^ key2;
  key1=tmp1;
  key2=tmp2;
  rnd[0]=key1;
  rnd[1]=key2;

  key1=key1table[key1];
  key2=key2table[key2];
  tmp1=key1 + key2;
  tmp2=key1 ^ key2;
  key1=tmp1;
  key2=tmp2;
  rnd[2]=key1;
  rnd[3]=key2;
```

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

  text1 =text1table[text1]+rnd[0];
  text2 =text2table[text2]^rnd[1];
  text1 =text1table[text1]+rnd[2];
  text2 =text2table[text2]^rnd[3];
  text1 =text1table[text1]+rnd[4];
  text2 =text2table[text2]^rnd[5];
  text1 =text1table[text1]+rnd[6];
  text2 =text2table[text2]^rnd[7];
  text1 =text1table[text1]+rnd[8];
  text2 =text2table[text2]^rnd[9];
  text1 =text1table[text1]+rnd[10];
  text2 =text2table[text2]^rnd[11];
  text1 =text1table[text1]+rnd[12];
  text2 =text2table[text2]^rnd[13];
  text1 =text1table[text1]+rnd[14];
  text2 =text2table[text2]^rnd[15];
  text1 =text1table[text1]+rnd[16];
  text2 =text2table[text2]^rnd[17];
  text1 =text1table[text1]+rnd[18];
  text2 =text2table[text2]^rnd[19];
  text[0]=text1;
  text[1]=text2;
}
```

key_gen 함수의 key1 = ROL(key1,4) 와 key2 = ROL(key2,6) 대신에 그에 해당되는 table을 미리 구하여 key1table과 key2table에 저장하였고, 마찬가지로 enc 함수의 ROL(text1,1) + ROL(text1,2) + ROL(text1,3) + ROL(text1,4) 와 ROL(text2,4) + ROL(text2,5) + ROL(text2,6) + ROL(text2,7)를 대신하여 text1table과 text2table에 저장하여 연산 과정을 최소화하였습니다. **key_gen과 enc에 들어있는 모든 for문을 없앴습니다.** 아두이노 코드로는 for을 사용하면 간단하지만 어셈블리 영역에서 for문을 사용하려면 i에 숫자를 대입하고 그 숫자를 다시 배열에 숫자에 해당되는 과정을 하는 것이 연산 시간을 더 걸리기 때문입니다.

2. 테스트 벡터 및 결과

```
void test_vector(){
  u8 key[2]={0x12,0x34};
  u8 text[2]={0x56,0x78};
  u8 rnd[20];

  key_gen(rnd,key);
  enc(text,rnd);
  Serial.println(text[0],HEX);
  Serial.println(text[1],HEX);
}
```

테스트 벡터를 위한 함수는 다음과 왼쪽 코드를 사용하였습니다.
test_vector(); 호출시 출력되는 값이 아래와 같음을 확인할 수 있습니다.
모든 방법들이 테스트 벡터를 통과하였고, 같은 결과를 출력하기 때문에 한 개만 캡처하였습니다.

3. 벤치마크 결과

첫 번째, 세 번째의 각 프로젝트 이름은 method1, method2로 명명하였습니다.

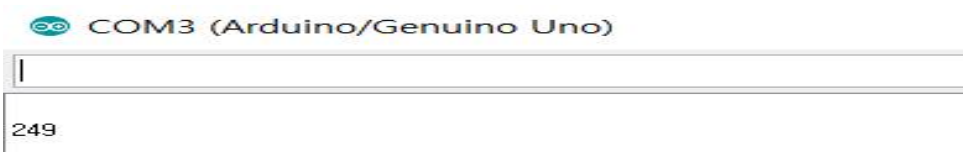
```
u32 time1;
u32 time2;

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

벤치마크의 코드는 다음과 같습니다.
10000번의 key_gen와 enc 함수를 호출하고 그 시간을 계산하였습니다.

1. method1의 벤치마크 결과

2. method2의 벤치마크 결과



벤치마크 결과 예상한대로 method2이 가장 빨랐으며, 기존 코드보다 45% 향상된 속도의 코드를 얻을 수 있었습니다.