2019102652 유동민

[XTS_AES]

```
int xts_aes(const void *in, void *out, size_t length, const void
{
  uint8_t P1[BLOCKLEN];
  uint8_t P2[BLOCKLEN];
  int m = length/BLOCKLEN;
  int r = length%BLOCKLEN;
  for(int j = 0 ; j < m ; j++){
    memcpy(P1, (uint8_t *)in+(j*16), BLOCKLEN);
    if(r && j == m-1){
      if(mode==ENCRYPT){
        xts_aes_block(P1,key,unit,j,mode);
        memcpy((uint8_t *)out+((j+1)*16), P1, r);
        memcpy(P2, (uint8_t *)in+((j+1)*16), r);
        memcpy(P2+r, P1+r, BLOCKLEN-r);
        xts_aes_block(P2,key,unit,j+1,mode);
        memcpy((uint8_t *)out+(j*16), P2, BLOCKLEN);
      }
      else if(mode==DECRYPT){
        xts_aes_block(P1,key,unit,j+1,mode);
        memcpy((uint8_t *)out+((j+1)*16), P1, r);
        memcpy(P2, (uint8_t *)in+((j+1)*16), r);
        memcpy(P2+r, P1+r, BLOCKLEN-r);
        xts_aes_block(P2,key,unit,j,mode);
        memcpy((uint8_t *)out+(j*16), P2, BLOCKLEN);
      }
    }
    else{
      xts_aes_block(P1,key,unit,j,mode);
      memcpy((uint8_t *)out+(j*16), P1, BLOCKLEN);
    }
  }

}
```

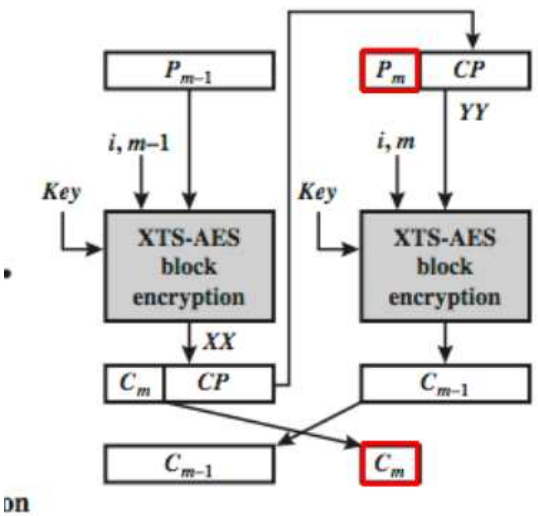
```
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44
---
c4 54 18 5e 6a 16 93 6e 39 33 40 38 ac ef 83 8b
fb 18 6f ff 74 80 ad c4 28 93 82 ec d6 d3 94 f0
---
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44
XTS-AES verification successful
---
6b e1 18 0a 53 2f 22 df 43 a7 18 31 21 f9 02 13
c4 54 18 5e
---
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44
44 44 44 44
XTS-AES CTS verification successful
---
c4 54 18 5e 6a 16 93 6e 39 33 40 38 ac ef 83 8b
fb 18 6f ff 74 80 ad c4 28 93 82 ec d6 d3 94 f0
---
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44
44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44
XTS-AES block verification successful
---
Random testing.............No error found
```

코드설명:

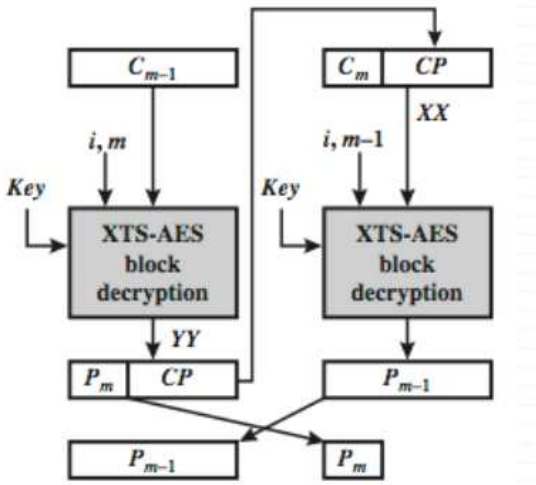
P1, P2: 과정을 출력할 버퍼

m, r: 블락의 개수와 마지막 블락이 BLOCKLEN이 아닐 경우 그 나머지값을 가져오는 변수

설명: 블락크기만큼 반복하며 암복호화 하며, 나머지 값이 있을 경우 마지막 두 블록을 예외처리하여 XTS-AES 알고리즘에 대응하여 암복호화합니다.



[암호화]



[복호화]

[XTS_AES_BLOCK]

```
void xts_aes_block(void *state, const void *key, unsigned long i, unsigned l
{

    KeyExpansion((uint8_t *)key, key1);
    KeyExpansion((uint8_t *)key+16, key2);
    for (int c=0; c<BLOCKLEN; c++){
      tmp[c] = (uint8_t) (i & 0xFF);
      i = i >> 8;
    }
    Cipher(tmp, key2, ENCRYPT);

    for(int c = 0 ; c < j+1; c++){
      if(c==0){
        memcpy(T,tmp,BLOCKLEN);
      }
      else{
        *((uint32_t *)T) = XTIME(*((uint32_t *)T));
        *((uint32_t *)T+1) = XTIME(*((uint32_t *)T+1));
        if (T[11] >> 7 & 1){T[4]++;}
        *((uint32_t *)T+2) = XTIME(*((uint32_t *)T+2));
        if (T[15] >> 7 & 1){T[8]++;}
        *((uint32_t *)T+3) = XTIME(*((uint32_t *)T+3));
        if (T[19] >> 7 & 1){T[12]++;}
        T[0] ^= 0x87;
      }
    }
```

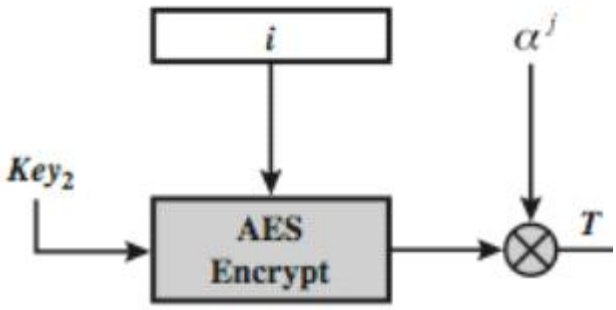
코드설명: TWEAK값을 생성하는 루틴입니다.

1) KeyExpansion을 통해 두 개의 키를 생성합니다.

2) unsigned long은 4바이트의 크기를 가지기 때문에 비트연산을 통해 tmp배열의 0x3333333333 (5바이트의 unit)의 값을 가져옵니다.

3) unit의 값과 key2를 AES알고리즘을 통해 암호화합니다.

4) j가 0일 경우 TWEAK값은 AES알고리즘을 통해 나온 사이퍼값입니다. 또한 j가 1씩 증가할수록 XTIME 매크로함수를 사용해 2만큼 곱해줍니다. 그러나 코드에서 32비트의 word로 쪼개어 왼쪽으로 쉬프트하기 때문에 4번째 8번째 12번째의 쉬프트를 검증하여 1을 더해줍니다. 마지막으로 x2+x+1을 모듈러 해줍니다.



[XTS_AES_BLOCK]

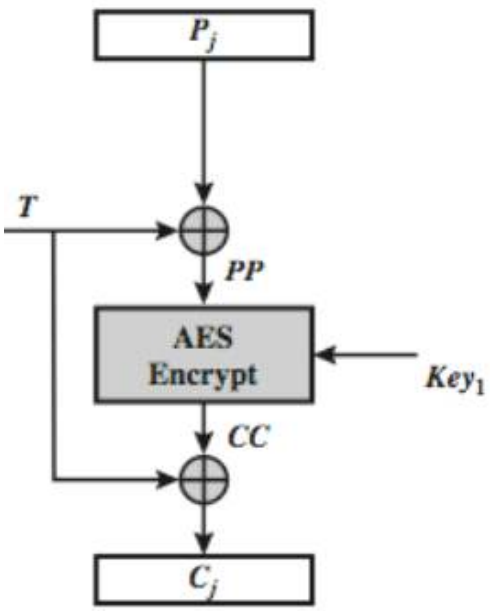
```
if(mode==ENCRYPT){
  for(int c = 0; c < BLOCKLEN; c++){
    XX[c] = T[c] ^ *((uint8_t *)state+c);
  }
  Cipher(XX, key1, mode);
  for(int c = 0; c < BLOCKLEN; c++){
    *((uint8_t *)state+c) = XX[c] ^ T[c];
  }
}// end ENCRYPT

else if(mode==DECRYPT){
  for(int c = 0; c < BLOCKLEN; c++){
    XX[c] = T[c] ^ *((uint8_t *)state+c);
  }
  Cipher(XX, key1, mode);
  for(int c = 0; c < BLOCKLEN; c++){
    *((uint8_t *)state+c) = XX[c] ^ T[c];
  }
}// end DECRYPT
```

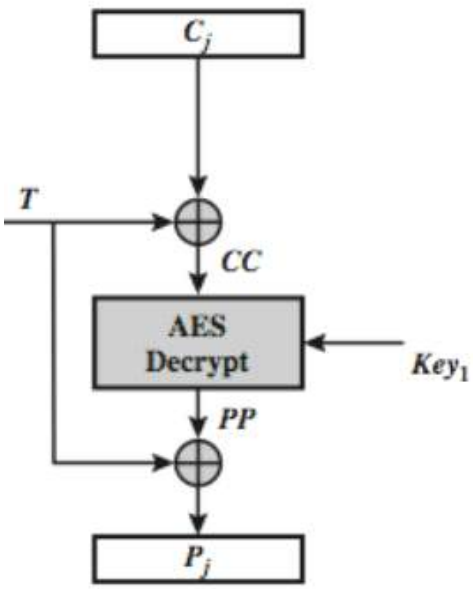
코드 설명: 암복호화 과정입니다.

ENC: TWEAK값과 평문을 xor해준 뒤 key1과 AES알고리즘을 통해 암호화합니다. 암호화결과인 XX값을 TWEAK값과 xor하여 암호화를 합니다.

DEC: TWEAK값과 암호문을 xor해준 뒤 key1과 AES알고리즘을 통해 복호화합니다. 복호화결과인 XX값을 TWEAK값과 xor하여 복호화를 합니다.



[암호화]



[복호화]