2020 암호분석경진대회

3번 문제 : 블록 암호

다음과 같이 수집된 암호문 C에 대응하는 평문 P를 찾으려고 한다. 수집된 암호문은 [그림 1]과 같이 정의된 블록암호 운영모드 암호화 과정을 통해 생성되었다. 이 때, E_K는 비밀키 K를 사용한 ARIA-128 암호화 과정을 의미한다. 블록암호 운영모드 암호화 과정에서 사용된 초기화벡터 IV 및 패딩 기법[그림 2]은 다음과 같다.

C: C9A735F6AC21EA3180E91C330F825D46B185CD404FF00987FDEB74DA6DC5F2948F24FC5B17547094E57E
F3547F593FF95774336D1AC41D4E5DA1BA54395EA42FFB66054387DA8D2DEAF11BAFD81EF4AACF7B59A8F72
7F1957BBC56595303EB2F2220537C9BEF67BF946BAA4E1460142D0F681086636828A2707305924B36BECDBE615
59A446107A30B243228E0FC85578E11A860CA32D96868B37DAC30328272AD8A10AA15C463630A20D7E82B1D6
7CFD84E9E12B62830CACE92F27F82CD3A1BA5CDCCE0E4AFD2B1AABA2ADFA7B059749BCAC5BDC38EAEA76
AF48C2F841629E18EAD5E88958356EF48078D099F3F66B95F4825A30AE8A658684C1B96007A42241DD325B3E0
1E9B2809CC2E92478C9FEDECE86753A4AF919B2D25FE079941CF7FD80EAD6E821B7AE3B9285C40888CE9D4D
505260B7DEE45D3B0893BBF2957D1B66F148020FA2CE4A0FB5CE8B2BD8457AD2142297CF1C3301618B26F3A4
B5D768A967864CE1635DCCB2E7B2C61B397472CCDA2771FDEFC876E82D79D55A5B9715BA8215C40D9F51AFE
7CCD6CB9AE52E097D65485AEFE730714A4E18CA4DBA67E3298EC1D4A7BFAD627A14CC25228DE1A39E5BF25
4BCAC898888BBC613873E9C1368A27C8B018EC1E00EF90B17E59B196D1

IV: 22496E697469616C20566563746F7222

입력: 평문 P
출력: 암호문 C

1. $P = \{P_0, P_1, P_2, \dots, P_{q-1}\}$, **$q$는 입력된 평문 블록 수**
2. $G_0 = IV$
3. **for** i **in** $(0, q-1)$:
4. $\quad C_i = E_K(P_i \oplus G_i)$
5. $\quad G_{i+1} = P_i \oplus C_i$
6. **return** $C = \{C_0, C_1, C_2, \dots, C_{q-1}\}$

[그림 1] 블록암호 운영모드 암호화 과정

입력: 평문 P
출력: 패딩이 수행된 평문 P^*

1. $P = \{P_0, P_1, P_2, \dots, P_{t-1}\}$, **$t$는 입력된 평문 블록 수**
2. $n = 16$
3. **if** $(ByteLength(P_{t-1}) < n)$ **{**
4. $\quad P_{t-1} = P_{t-1} || 80 || 00^{n-ByteLength(P_{t-1})-1}$
5. $\quad P^* = \{P_0, P_1, P_2, \dots, P_{t-1}\}$ **}**
6. **else** $(ByteLength(P_{t-1}) = n)$ **{**
7. $\quad P_q = 80 || 00^{n-1}$
8. $\quad P^* = \{P_0, P_1, P_2, \dots, P_{t-1}, P_t\}$ **}**
9. **return** P^*

[그림 2] 패딩 수행 과정

다음 첨부된 라이브러리 파일은 제시된 블록암호 운영모드에 대한 복호화를 수행한 후 평문에 적용된 패딩 패턴을 검사하는 API를 포함하고 있다. 해당 API를 사용하면 입력된 암호문에 대해 복호화된 결과에 대한 패딩 형식이 올바른지에 대한 판별 결과를 얻을 수 있다. 반환 값이 0: VALID 이면 패딩 형식이 올바름을 의미하고, 1: INVALID 이면 패딩 형식이 올바르지 않음을 의미한다. 첨부된 라이브러리 파일 내에 포함된 API 사용하여 암호문 C에 대응되는 평문 P를 구하시오. (단, 구한 평문 값에 대응되는 문자는 ASCII 코드를 활용하여 알파벳으로 변환하시오.)

Dll 파일에 대한 설명은 다음을 참고하시오.

※ 첨부된 라이브러리 파일 설명

첨부된 Decryptor.zip은 “Decryptor.dll”, “Decryptor.lib”, “decryptor.h”로 구성되어 있으며 각각에 대한 설명은 다음과 같다.

- Decryptor.dll: 블록암호 운영모드 복호화 및 패딩 패턴 확인 모듈을 포함하는 동적 라이브러리 파일
- Decryptor.lib: 프로그램 실행시 참조 라이브러리 파일
- decryptor.h: 라이브러리 파일 import 후 실행파일 생성 시 필요한 참조 헤더파일

Decryptor.dll에 포함된 복호화 모듈은 암호문 및 초기화벡터를 포함하는 CTX 구조체를 파라미터로 사용하며 CTX 구조체는 다음과 같이 구성됨.

```
struct CTX  {
    unsigned char* ciphertext; // 암호문
    int ciphertext_length; // 암호문의 Byte 길이
    unsigned char* IV; // 초기화벡터(16 Bytes)  };
```

복호화 및 패딩 패턴 확인 모듈 API 프로토타입은 아래와 같으며 입력된 암호문에 대해 복호화를 수행한 후, 복호화 결과에 대해 올바른 패딩 형식인지를 검사하고 그 결과를 반환한다.

```
int Dec_CTX(struct CTX* ctx);
```

함수 반환 값
- 0: VALID (복호화 결과의 패딩 형식이 올바름)
- 1: INVALID (복호화 결과의 패딩 형식이 올바르지 않음)

- API 사용 예시 코드

```
#include <iostream>
#include "decryptor.h"

int main()
{
    Byte IV[16] = { 0x22, 0x49, 0x6e, 0x69, 0x74, 0x69, 0x61, 0x6c, 0x20, 0x56, 0x65, 0x63, 0x74, 0x6f, 0x72, 0x22 };
    Byte c1[] = {
        0xC9, 0xA7, 0x35, 0xF6, 0xAC, 0x21, 0xEA, 0x31, 0x80, 0xE9, 0x1C, 0x33, 0x0F, 0x82, 0x5D, 0x46,
        0xB1, 0x85, 0xCD, 0x40, 0x4F, 0xF0, 0x09, 0x87, 0xFD, 0xEB, 0x74, 0xDA, 0x6D, 0xC5, 0xF2, 0x94,
        0x8F, 0x24, 0xFC, 0x5B, 0x17, 0x54, 0x70, 0x94, 0xE5, 0x7E, 0xF3, 0x54, 0x7F, 0x59, 0x3F, 0xF9,
        0x57, 0x74, 0x33, 0x6D, 0x1A, 0xC4, 0x1D, 0x4E, 0x5D, 0xA1, 0xBA, 0x54, 0x39, 0x5E, 0xA4, 0x2F,
        0xFB, 0x66, 0x05, 0x43, 0x87, 0xDA, 0x8D, 0x2D, 0xEA, 0xF1, 0x1B, 0xAF, 0xD8, 0x1E, 0xF4, 0xAA,
        0xCF, 0x7B, 0x59, 0xA8, 0xF7, 0x27, 0xF1, 0x95, 0x7B, 0xBC, 0x56, 0x59, 0x53, 0x03, 0xEB, 0x2F,
        0x22, 0x20, 0x53, 0x7C, 0x9B, 0xEF, 0x67, 0xBF, 0x94, 0x6B, 0xAA, 0x4E, 0x14, 0x60, 0x14, 0x2D,
        0x0F, 0x68, 0x10, 0x86, 0x63, 0x68, 0x28, 0xA2, 0x70, 0x73, 0x05, 0x92, 0x4B, 0x36, 0xBE, 0xCD,
        0xBE, 0x61, 0x55, 0x9A, 0x44, 0x61, 0x07, 0xA3, 0x0B, 0x24, 0x32, 0x28, 0xE0, 0xFC, 0x85, 0x57,
        0x8E, 0x11, 0xA8, 0x60, 0xCA, 0x32, 0xD9, 0x68, 0x68, 0xB3, 0x7D, 0xAC, 0x30, 0x32, 0x82, 0x72,
        0xAD, 0x8A, 0x10, 0xAA, 0x15, 0xC4, 0x63, 0x63, 0x0A, 0x20, 0xD7, 0xE8, 0x2B, 0x1D, 0x67, 0xCF,
        0xD8, 0x4E, 0x9E, 0x12, 0xB6, 0x28, 0x30, 0xCA, 0xCE, 0x92, 0xF2, 0x7F, 0x82, 0xCD, 0x3A, 0x1B,
        0xA5, 0xCD, 0xCC, 0xE0, 0xE4, 0xAF, 0xD2, 0xB1, 0xAA, 0xBA, 0x2A, 0xDF, 0xA7, 0xB0, 0x59, 0x74,
        0x9B, 0xCA, 0xC5, 0xBD, 0xC3, 0x8E, 0xAE, 0xA7, 0x6A, 0xF4, 0x8C, 0x2F, 0x84, 0x16, 0x29, 0xE1,
        0x8E, 0xAD, 0x5E, 0x88, 0x95, 0x83, 0x56, 0xEF, 0x48, 0x07, 0x8D, 0x09, 0x9F, 0x3F, 0x66, 0xB9,
        0x5F, 0x48, 0x25, 0xA3, 0x0A, 0xE8, 0xA6, 0x58, 0x68, 0x4C, 0x1B, 0x96, 0x00, 0x7A, 0x42, 0x24,
        0x1D, 0xD3, 0x25, 0xB3, 0xE0, 0x1E, 0x9B, 0x28, 0x09, 0xCC, 0x2E, 0x92, 0x47, 0x8C, 0x9F, 0xED,
        0xEC, 0xE8, 0x67, 0x53, 0xA4, 0xAF, 0x91, 0x9B, 0x2D, 0x25, 0xFE, 0x07, 0x99, 0x41, 0xCF, 0x7F,
        0xD8, 0x0E, 0xAD, 0x6E, 0x82, 0x1B, 0x7A, 0xE3, 0xB9, 0x28, 0x5C, 0x40, 0x88, 0x8C, 0xE9, 0xD4,
        0xD5, 0x05, 0x26, 0x0B, 0x7D, 0xEE, 0x45, 0xD3, 0xB0, 0x89, 0x3B, 0xBF, 0x29, 0x57, 0xD1, 0xB6,
        0x6F, 0x14, 0x80, 0x20, 0xFA, 0x2C, 0xE4, 0xA0, 0xFB, 0x5C, 0xE8, 0xB2, 0xBD, 0x84, 0x57, 0xAD,
        0x21, 0x42, 0x29, 0x7C, 0xF1, 0xC3, 0x30, 0x16, 0x18, 0xB2, 0x6F, 0x3A, 0x4B, 0x5D, 0x76, 0x8A,
        0x96, 0x78, 0x64, 0xCE, 0x16, 0x35, 0xDC, 0xCB, 0x2E, 0x7B, 0x2C, 0x61, 0xB3, 0x97, 0x47, 0x2C,
        0xCD, 0xA2, 0x77, 0x1F, 0xDE, 0xFC, 0x87, 0x6E, 0x82, 0xD7, 0x9D, 0x55, 0xA5, 0xB9, 0x71, 0x5B,
        0xA8, 0x21, 0x5C, 0x40, 0xD9, 0xF5, 0x1A, 0xFE, 0x7C, 0xCD, 0x6C, 0xB9, 0xAE, 0x52, 0xE0, 0x97,
        0xD6, 0x54, 0x85, 0xAE, 0xFE, 0x73, 0x07, 0x14, 0xA4, 0xE1, 0x8C, 0xA4, 0xDB, 0xA6, 0x7E, 0x32,
        0x98, 0xEC, 0x1D, 0x4A, 0x7B, 0xFA, 0xD6, 0x27, 0xA1, 0x4C, 0xC2, 0x52, 0x28, 0xDE, 0x1A, 0x39,
        0xE5, 0xBF, 0x25, 0x4B, 0xCA, 0xC8, 0x98, 0x88, 0x8B, 0xBC, 0x61, 0x38, 0x73, 0xE9, 0xC1, 0x36,
        0x8A, 0x27, 0xC8, 0xB0, 0x18, 0xEC, 0x1E, 0x00, 0xEF, 0x90, 0xB1, 0x7E, 0x59, 0xB1, 0x96, 0xD1
    };

    int ret;

    struct CTX ctx1;

    ctx1.ciphertext = c1;
    ctx1.cipher_length = sizeof(c1);
    ctx1.IV = IV;

    ret = Dec_CTX(&ctx1);

    if (ret == 0)
        printf("VALID₩n");
    else
    {
        printf("INVALID₩n");
    }


    return 0;
}
```

- 실행화면