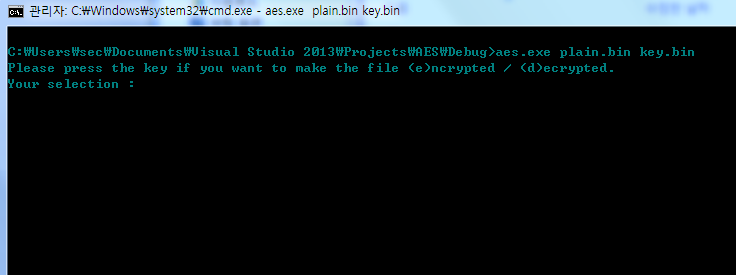
|  |  |
| --- | --- |
|  | **CSE4308 -002** |
|  | 컴퓨터정보공학과  12131603  조경민  drzix@naver.com  01031772679 |

|  |
| --- |
| **[AES 구현]** |
| 과제의 요구사항에 맞추어 AES를 구현하였다. |

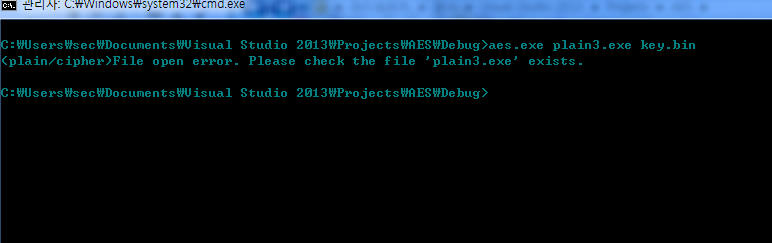


1. 프로그램 실행 방법과 예시

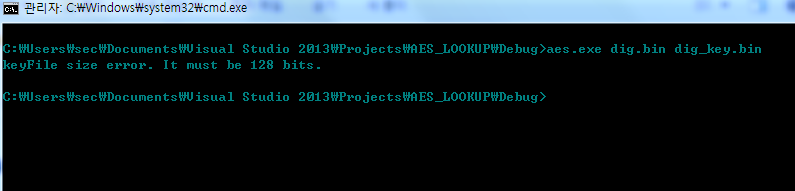
프로그램은 기본적으로 커맨드 창에서 실행된다. 3개의 argument로 이루어지는데 가장 처음 프로그램의 이름을 입력하고 암, 복호화 하고자 하는 텍스트 파일의 경로, 마지막으로 키 파일의 경로를 입력하여 준다. ‘e’ 키(대소문자 무관)를 누르면 암호화가 되고, ‘d’ 키를 누르면 복호화가 된다.



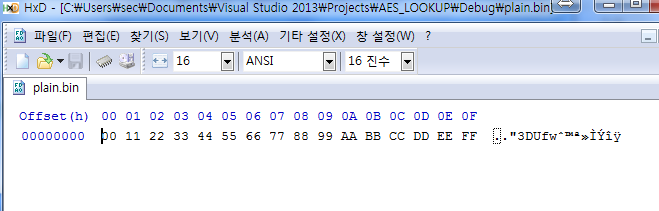
만약 경로가 잘못되어서 파일을 찾을 수 없을 경우 아래와 같은 오류를 접할 수 있다.

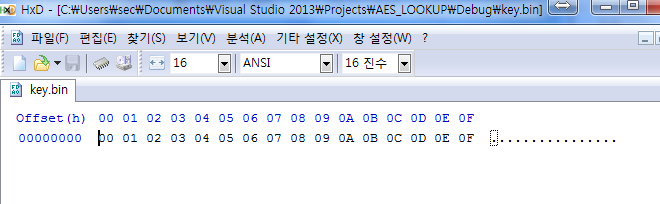


이 때, 텍스트 파일은 반드시 16바이트의 배수여야 하고, 키 파일은 16바이트여야 한다. 그렇지 않은 경우, 아래와 같은 오류가 뜬다.

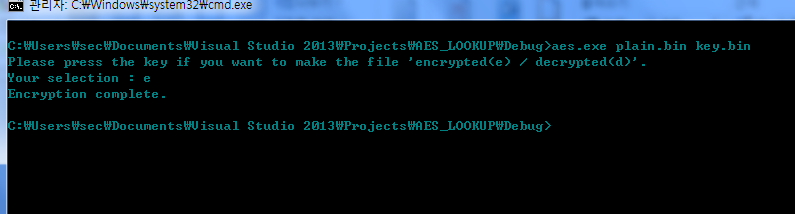


다음은 텍스트 파일 ‘plain.bin’와 키 파일 ‘key.bin’이다.

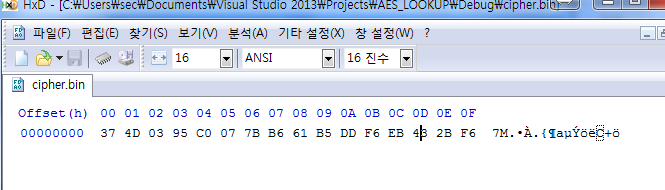




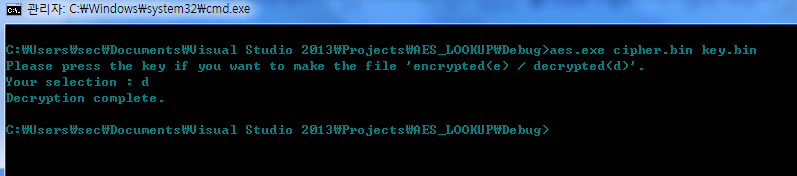
이를 이용하여 암호화하는 화면은 아래와 같다.



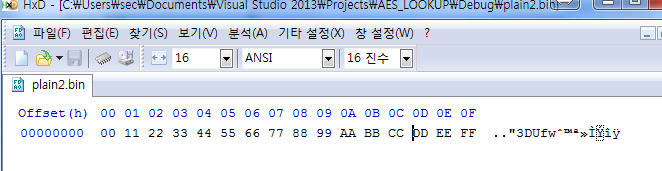
암호화된 파일은 ‘aes.exe’가 존재하는 디렉토리에 ‘cipher.bin’으로 출력된다.



다시 ‘cipher.bin’ 파일을 같은 키를 이용하여 복호화하여 본다.



복호화된 파일은 ‘aes.exe’가 존재하는 디렉토리에 ‘plain2.bin’으로 출력된다.



1. 소스 코드

본 프로그램은 크게 다음과 같이 구성되어 있다.

**AES 암∙복호화 알고리즘을 담당하는 aes.h (선언부)와 aes.c (구현부)**

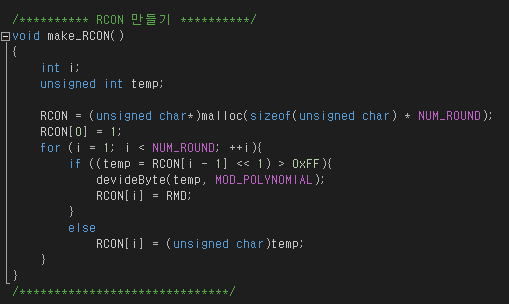
**Galois Field 내의 연산을 규정하는 galoisField.h (선언부)와 galoisField.c (구현부)**

**파일 입출력을 담당하는 fileHandle.h (선언부)와 fileHandle.c (구현부)**

**그리고 프로그램의 실행 부분인 main.c**

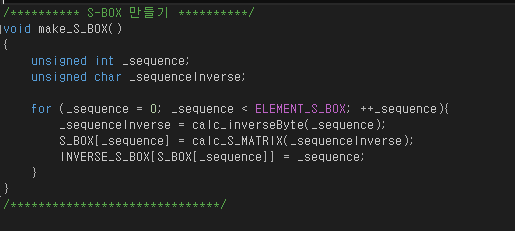
* 1. **aes.c 의 함수들**

본격적으로 AES 알고리즘을 진행하기 전에 키 확장을 위한 RCON과 바이트 치환을 위한 S-BOX가 필요하다. 다음은 RCON을 만드는 함수이다.



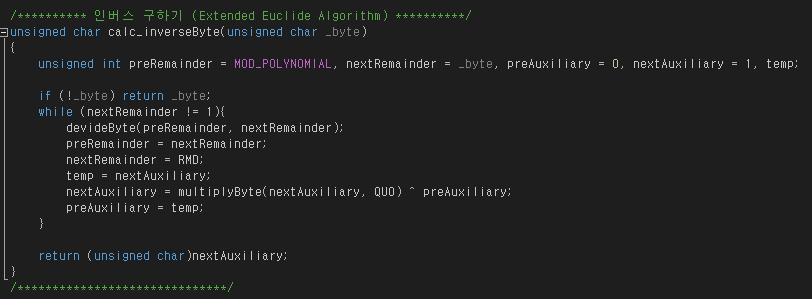
1을 2씩 곱해주고 8비트가 넘어가면 irreducible polinomial로 모듈러 연산을 해준다.

아래의 함수는 S-BOX를 제작한다.



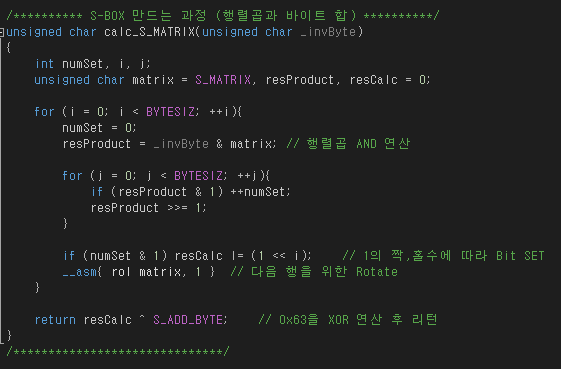
S-BOX를 시각화하면 2차원 표로 볼 수 있지만, 프로그램 내부적으로는 일차원 배열로 충분해도 관계가 없다. S-BOX를 만듦과 동시에 Inverse S-BOX도 제작한다.

한편 주어진 바이트의 Inverse를 구하는 함수는 아래와 같다.



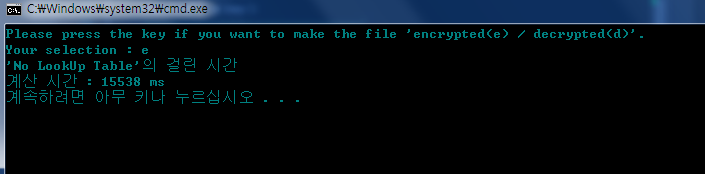
전체적으로 extended euclide algorithm을 진행한다.

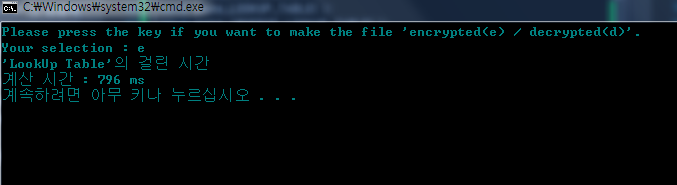
구해진 inverse 바이트를 이용하여 S-BOX에 적용되는 행렬과의 연산은 다음과 같다.



각 행과 inverse 바이트를 &연산한 후 0x63 바이트와 xor 연산한 값을 리턴한다.

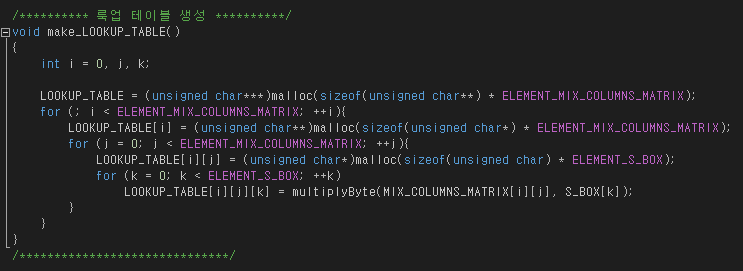
S-BOX를 구했으니 이와 Mix Columns를 할 때 쓰이는 행렬과 합쳐 Lookup Table을 제작해야 한다. 굳이 제작하지 않아도 결과를 구하는데 상관은 없지만 그 차이는 아래의 스크린샷을 통해 느낄 수 있다.





1MB 크기의 텍스트 파일을 암호화한 결과이다. Lookup Table이 있을 때와 없을 때의 연산 시간의 차이는 엄청나다!

때문에 반드시 Lookup Table을 만들어야 한다.



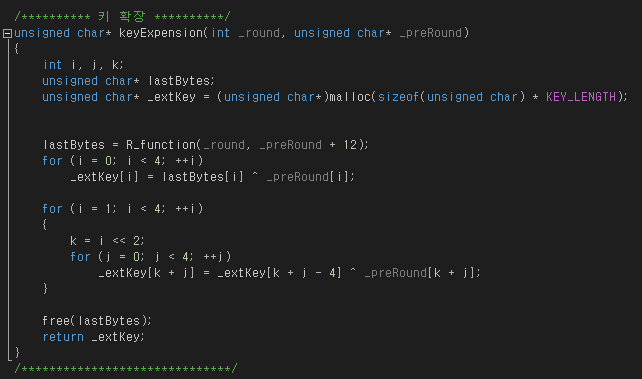
삼중 반복문을 통해 Mix Columns 시 필요한 행렬과 S-BOX의 요소들을 일일이 계산하여 저장한다. 총 4096바이트의 저장공간을 요한다. 그러나 실제로 Mix Columns 행렬은 첫 행이 rotate된 행들로 이루어져있으므로 실제로 계산에 필요한 값은 01, 02, 03(inverse의 경우 0e, 0b, 0d, 09의 네가지)과 S-BOX 요소들과의 계산 결과들 뿐이다. 이를 토대로 필요한 저장공간을 줄일 수 있다. 그러나 코딩의 편리성과 논리가 지나치게 복잡해지는 것을 막기 위해 전자를 택하였다.

Lookup Table의 생성이 끝났다면 키를 미리 확장해 놓고 저장한다.



실제 라운드마다 확장해도 상관 없으나 복호화의 경우 가장 마지막으로 확장된 키부터 복호화가 시작되므로 확장한 값을 먼저 저장해놓는 방식을 취하였다. 키 확장이 끝나면 RCON은 더 이상 필요가 없으므로 할당 해제한다.

실제로 키를 확장하는 함수는 아래와 같다.

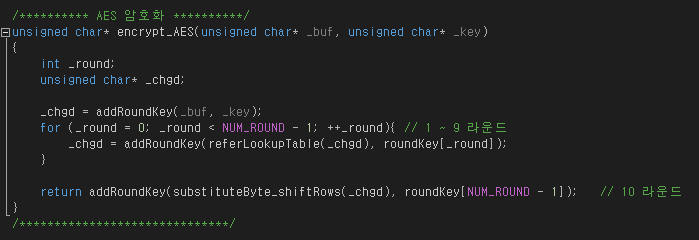


실제 AES Key expension과 방식이 같음을 직관적으로 알 수 있다.

한편 키 확장에서 마지막 네 바이트를 조작하는 R함수는 아래와 같다.

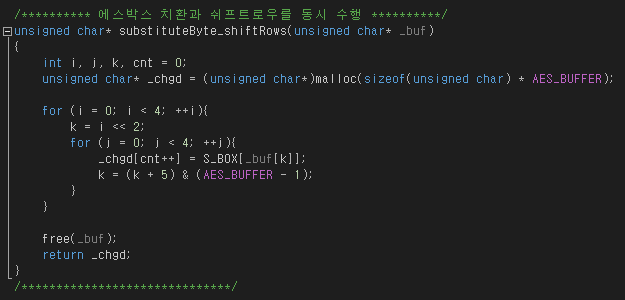


이제 키 확장도 끝났으니 전체적인 암호화 과정을 나타내는 함수를 보인다.



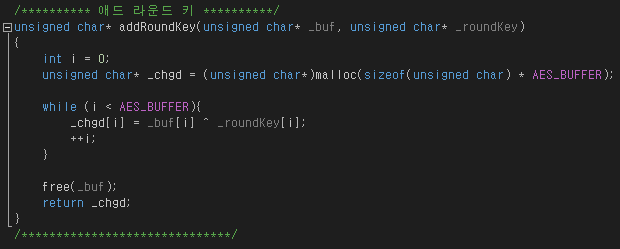
마지막 10라운드를 제외하고는 계속해서 Lookup Table을 참고한다.

10라운드에서 substitute byte와 shift rows 과정을 함께하는 것을 볼 수 있는데, 문자를 치환하고 위치를 바꾸나, 위치를 바꾼 후 치환을 하나 결과는 같다. 때문에 이를 동시에 수행하는 함수를 사용했다.



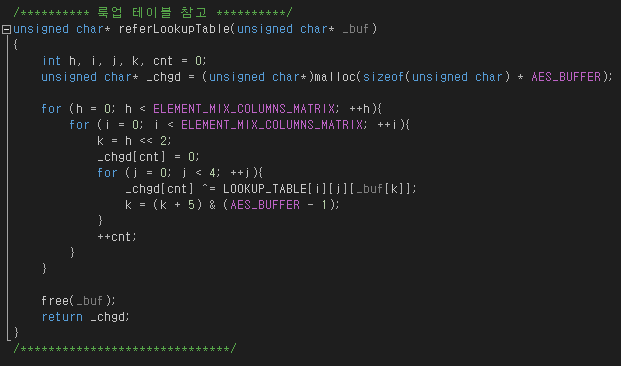
shift rows를 수행할 때 각 행 별로 rotate를 하는 것이 원칙이다. 그러나 프로그램 내부에서는 실제로 rotate를 하지 않고 rotate 전과 rotate 후의 위치 차이의 규칙을 이용하여 바로 넣어준다. 변수 k에 5를 더하고 모듈러 연산을 취하는 코드가 이에 해당한다. (이 때도 진짜 모듈러 연산을 사용하진 않는다. AES\_BUFFER가 16이라는 사실을 이용하여 0xF와 &연산을 함으로써 나머지를 구한다. 이렇게 비트연산을 이용하는 방법이 실제 모듈러 연산보다 훨씬 빠르고 효과적이다.)

add round key 함수는 아래와 같다.



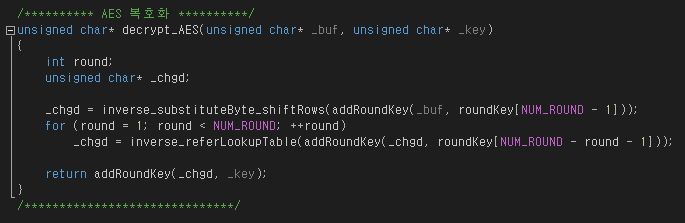
해당 라운드의 키와 텍스트를 바이트 별로 xor 연산해주는 과정이다.

Lookup table을 참고하는 함수는 아래와 같다.



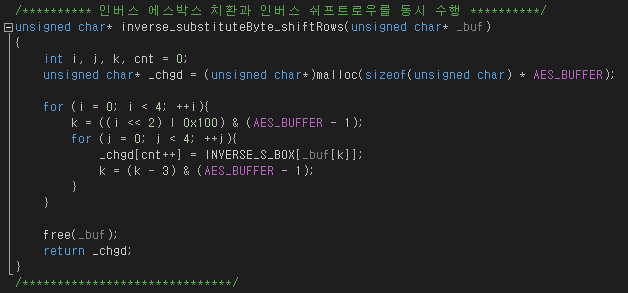
쉽게 말해 substitute byte, shift rows, mix colmns를 동시에 수행하는 것이나 다름 없다.

여태까지 암호화하는 함수들에 대해 알아보았다. 이제 복호화하는 함수들에 대해 알아보자. Inverse S-BOX를 구하는 것과 키 확장에 해당하는 암호화 과정에 쓰이는 함수를 그대로 쓴다. 아래는 전체적인 복호화 과정이다.



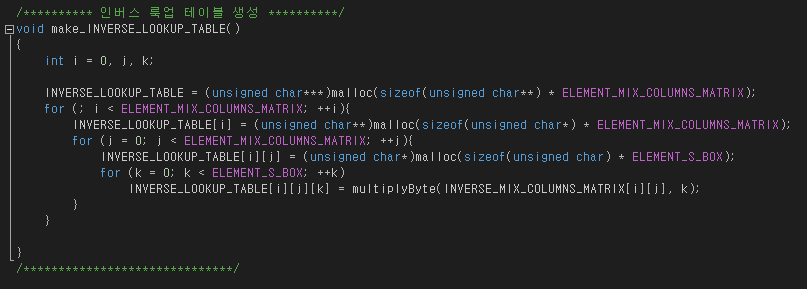
복호화 과정에서 순서과 암호화 과정과 다르기에 별도로 inverse Lookup Table과 inverse 함수들을 구현해야 한다.

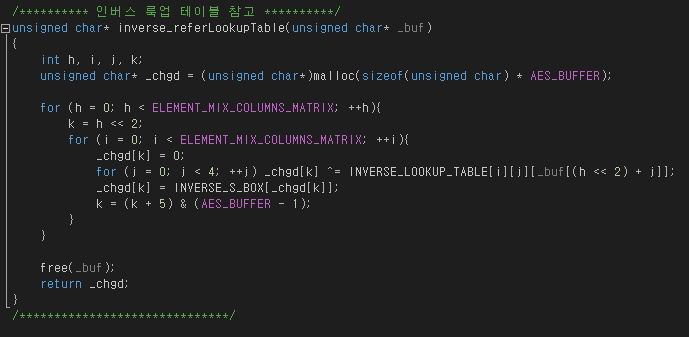
아래는 복호화 과정에서 치환과 행 변환을 역으로 수행하는 함수다.



복호화에서의 shift rows 역시 규칙을 이용하여 코딩하였다.

Inverse Lookup Table 생성과 이를 참고하는 함수들은 아래와 같다.

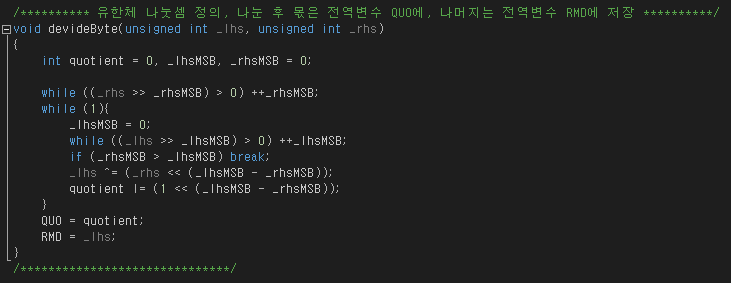




암호화할 때와 차이가 있다면 Lookup Table 생성 시 S-BOX를 참고하지 않는 다는 점이다. 이는 mix columns 한 후에 치환을 수행하기 때문이다.

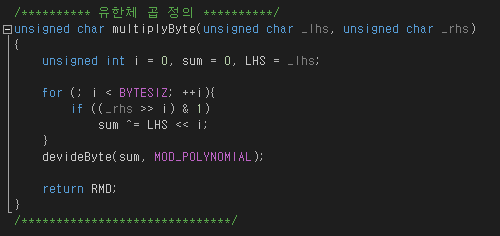
* 1. **galoisField.c 의 함수들**

아래는 유한체에서 나눗셈을 수행하는 함수다.



마치 어셈블리어에서 나누기를 수행하면 몫과 나머지가 각각 eax와 edx에 저장되듯이 위 함수는 몫과 나머지를 전역변수 QUO와 RMD에 저장한다.

다음은 유한체에서 곱셈을 수행하는 함수다.



곱의 결과를 irreducible polynomial으로 나누어 나머지를 리턴하는 방식이다.

fileHandle.c와 main.c는 기본적인 프로그램의 구성이다. 파일의 정상적인 실행이나 파일의 크기를 탐지하는 것들이므로 코드 분석은 생략한다.

1. 개발 환경

개발 도구 : 삼성 Series 7 Chronos (노트북)

개발 언어 : C

컴파일러 버전 : Microsoft Visual Studio professional 2013 (버전 12.0.30501.00 Update 2)

시스템 사양:

OS - Windows7 64bit

CPU - Intel i7-3615QM

RAM - 8GB