**변형된 AES-128 과제 보고서**

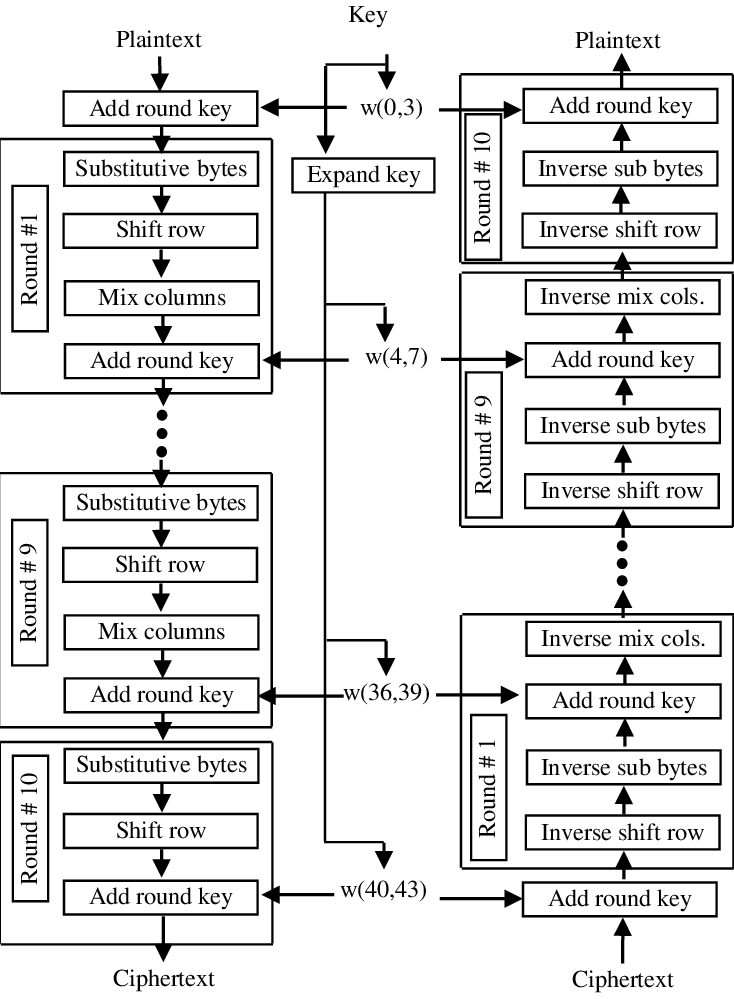
12161569 컴퓨터공학과

박도윤

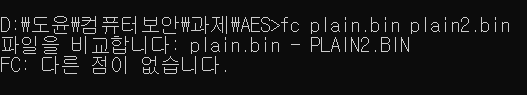
전화번호 : 010-9429-2169

이메일 : [12161569@inha.edu](mailto:12161569@inha.edu)

* **개요**

1. 동작 원리
2. 전체 과정
3. Add round key
4. Substitution bytes
5. Shift row
6. Mix column
7. Key expansion
8. 함수 목록
9. 사용된 변수형
10. 소스 코드
11. **동작 원리**
12. **전체 과정**

* [Encryption] 표준 입력 ‘e’를 입력 받으면, 16 바이트 key를 입력 받아서 확장한 후, 확장된 key의 맨 처음 부분을 XOR한 후에 [substitution bytes – shift row – mix column – add round key] round를 10번 돌도록 구현했습니다.
* 맨 마지막 round에서는 mix column 연산을 제외한 3개의 연산만 수행되도록 구현했습니다.
* [Decryption] 표준 입력 ‘d’를 입력 받으면, 마찬가지로 16 바이트 key를 입력 받아서 확장한 후, 확장된 key의 맨 마지막 부분과 XOR한 후에 [inverse shift row – inverse substitution bytes – add round key – inverse mix column] round를 10번 돌도록 구현했습니다.
* 맨 마지막 round에서는 encryption과 마찬가지로 mix column 연산을 제외한 3개의 연산만 수행되도록 구현했습니다.
* Encryption에서는 key의 대입이 w(0,3) ~ w(40,43) 순서로, Decryption에서는 반대로 w(40,43) ~ w(0,3) 순서로 진행되도록 구현했습니다.



* **aes.exe e**를 입력하면 plain.bin과 key.bin을 읽어 encryption을 한 후 cipher.bin을 출력하고, **aes.exe d**를 입력하면 cipher.bin과 key.bin을 읽어 decryption을 한 후 plain2.bin을 출력하도록 구현하였습니다.
* 각 round마다 중간 값을 출력하도록 구현하였습니다.
* Plain.bin과 plain2.bin 두 파일을 비교했을 때, 다른 점이 없는 것을 확인할 수 있습니다.
* 16 byte 크기의 key.bin 파일을 입력 받고, 16 byte의 n배 크기의 파일(plain.bin이나 cipher.bin)을 입력 받았을 때, plain.bin이나 cipher.bin 파일을 16byte씩 분할하여 확장된 key[w(0,3)~w(40,43)]를 이용해 encryption/decryption을 n번 실행하도록 구현하였습니다. (ECB 모드)

1. **Add round key**

* 단순 bit XOR 연산으로 구현하였습니다.

1. **Substitution bytes**

* 4개의 word를 인자로 받아, 한 word씩 각각 연산하도록 구현하였습니다.
* 한 word를 8bit씩 분리해서 각각 s-box 연산을 하도록 구현하였습니다.

* S-box 연산에서, 각 바이트의 비트들에 대하여 이러한 변환을 수행하도록 구현하였습니다.

1. **Shift row**

* 값을 배열에 담고 배열 값을 이동시킨 뒤, 다시 배열을 값으로 변환하는 방식으로 구현하였습니다.
* Shift row 함수 배열(tmp[16])의 예 (순방향)

☺ 입력 받은 인자 temp[4] = {**0xAB0518E4**, **0x8B403F4E**, **0x897FF02F**, **0x35F1FCC4**}

|  |  |  |  |
| --- | --- | --- | --- |
| Tmp[0] | Tmp[4] | Tmp[8] | Tmp[12] |
| Tmp[1] | Tmp[5] | Tmp[9] | Tmp[13] |
| Tmp[2] | Tmp[6] | Tmp[10] | Tmp[14] |
| Tmp[3] | Tmp[7] | Tmp[11] | Tmp[15] |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **AB** | **8B** | **89** | **35** |  | **AB** | **8B** | **89** | **35** |
| **05** | **40** | **7F** | **F1** |  | **40** | **7F** | **F1** | **05** |
| **18** | **3F** | **F0** | **FC** |  | **F0** | **FC** | **18** | **3F** |
| **E4** | **4E** | **2F** | **C4** |  | **C4** | **E4** | **4E** | **2F** |

1. **Mix column**

* 값을 4 \* 4 배열에 옮긴 뒤, 행렬 곱셈을 한 후 다시 배열을 값으로 변환하는 방식으로 구현하였습니다.
* 순 변환 행렬과 역 변환 행렬을 mix\_arr[4][4] 배열에 먼저 넣어준 뒤, 인자로 받은 값의 배열과 행렬 곱셈을 하는 방식으로 구현하였습니다.

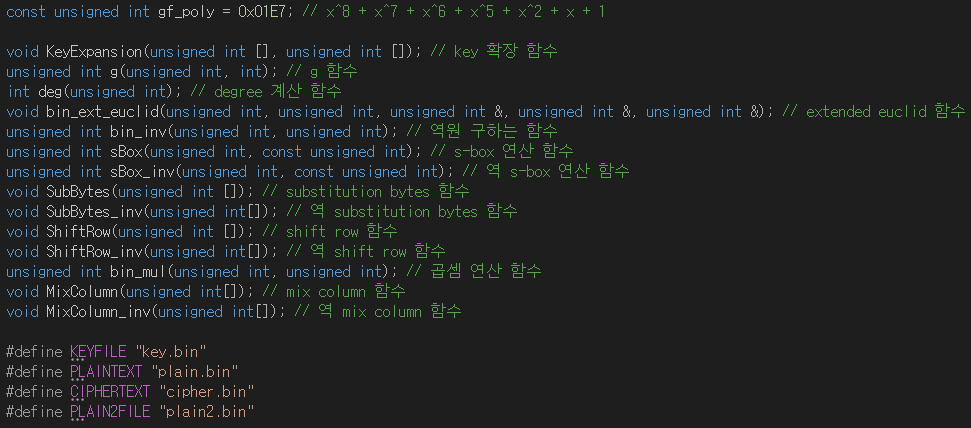
|  |  |  |  |
| --- | --- | --- | --- |
| 02 | 03 | 01 | 01 |
| 01 | 02 | 03 | 01 |
| 01 | 01 | 02 | 03 |
| 03 | 01 | 01 | 02 |

|  |  |  |  |
| --- | --- | --- | --- |
| 0E | 0B | 0D | 09 |
| 09 | 0E | 0B | 0D |
| 0D | 09 | 0E | 0B |
| 0B | 0D | 09 | 0E |

▲ 순 변환 행렬 ▲ 역 변환 행렬

1. **Key expansion**

* 입력 받은 key 16바이트를 44개의 word 배열로 확장하도록 구현하였습니다.
* g 함수는 word 한 개와 encryption 관점에서 몇 번째 round인지 나타내는 변수를 인자로 받아서 위치 이동과 s-box 연산, RC 값을 구한 뒤 XOR 연산을 하도록 구현하였습니다.
* RC 값은 인자로 받은 round 값을 확인한 후 각 round마다 다른 방식으로 계산되도록 구현하였습니다.

1. **함수 목록**

* 함수 목록과 irreducible polynomial, 입출력 파일 경로는 다음과 같습니다.

1. **사용된 변수형**

* 하나의 word로 unsigned int형을 사용하였습니다.
* 확장된 key는 unsigned int형 배열 key[44]로, encryption / decryption 중간의 상태 배열로 unsigned int형 temp[4]를 사용하였습니다.
* unsigned int형 변수 배열의 사용 예

☺ 주어진 key : 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

Unsigned int input\_key[4]

input\_key[0] → **0x00010203**

input\_key[1] → **0x04050607**

input\_key[2] → **0x08090A0B**

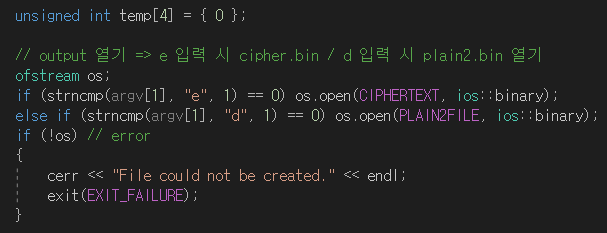
input\_key[3] → **0x0C0D0E0F**

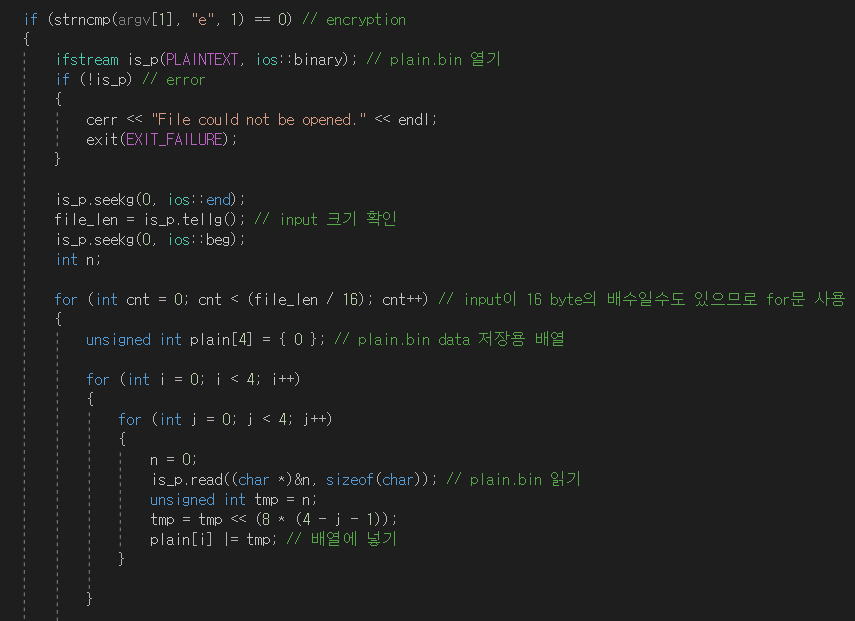
1. **소스 코드**
   * + Main 함수

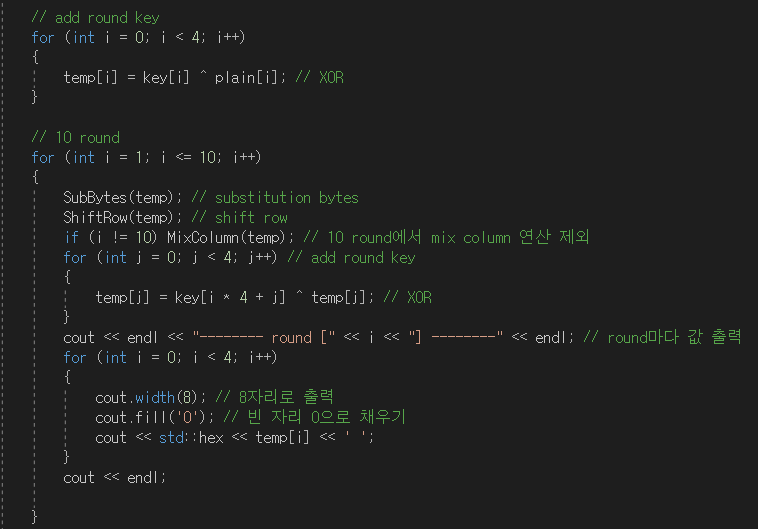
텍스트이(가) 표시된 사진

자동 생성된 설명

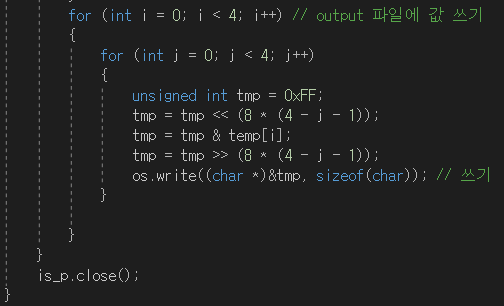
* Key.bin 파일을 연 후 input\_key 배열에 값을 저장하고 확장하는 코드입니다.
* 저장한 key 값을 KeyExpansion 함수를 이용해 확장하여 key 배열에 넣도록 구현하였습니다.



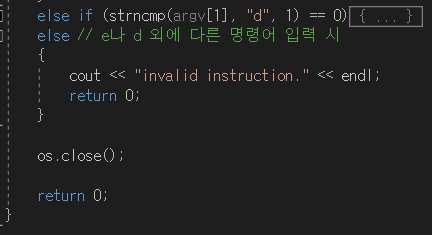
* Output 파일을 여는 코드입니다.
* Temp[4] 배열은 중간 상태를 나타내는 배열입니다.
* 표준 입력으로 ‘e’를 입력 받았을 때 plain.bin 파일을 열고 해당 파일을 읽어 plain[4] 배열에 저장하는 코드입니다.
* Plain.bin 파일의 크기를 계산하여 plain.bin 파일을 몇 개의 16 byte로 분할하여 encryption을 할 건지 for문을 이용해 정하도록 구현하였습니다.



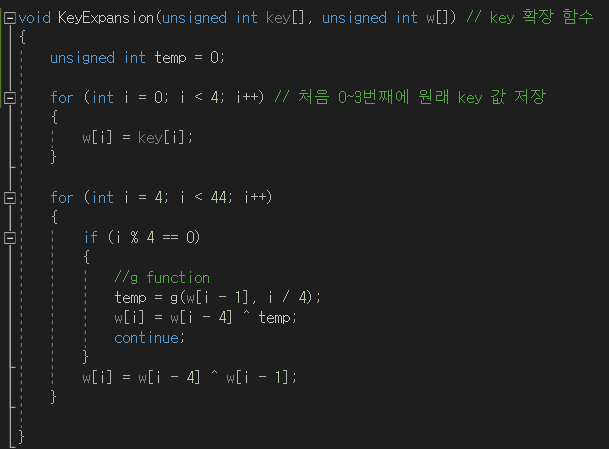
* Add round key를 XOR 연산으로 수행한 후, 10번의 round를 도는 코드입니다.
* 각 round마다 [substitution bytes – shift row – mix column – add round key] 연산을 하고 그 중간 결과값을 출력하고, 맨 마지막 10 round에서 mix column 연산을 제외하도록 구현하였습니다.



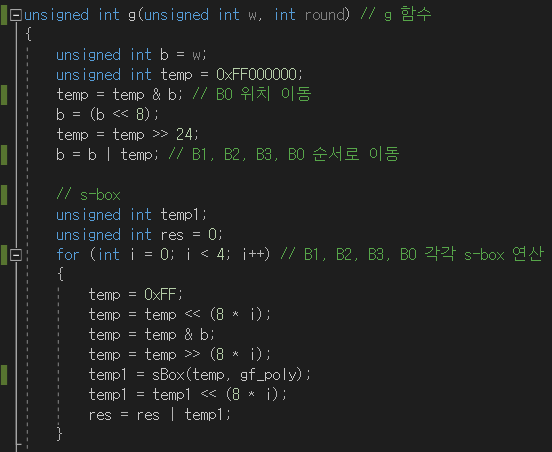
* 10 round를 돈 후 output 파일에 결과값을 쓰는 코드입니다.

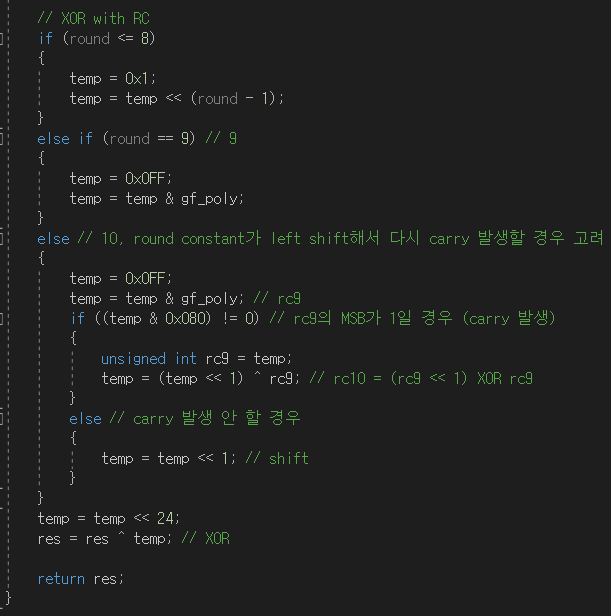
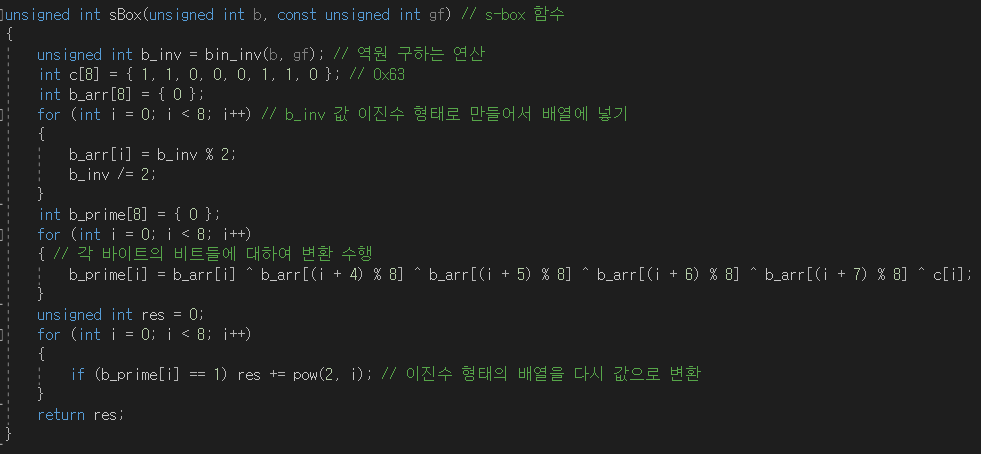


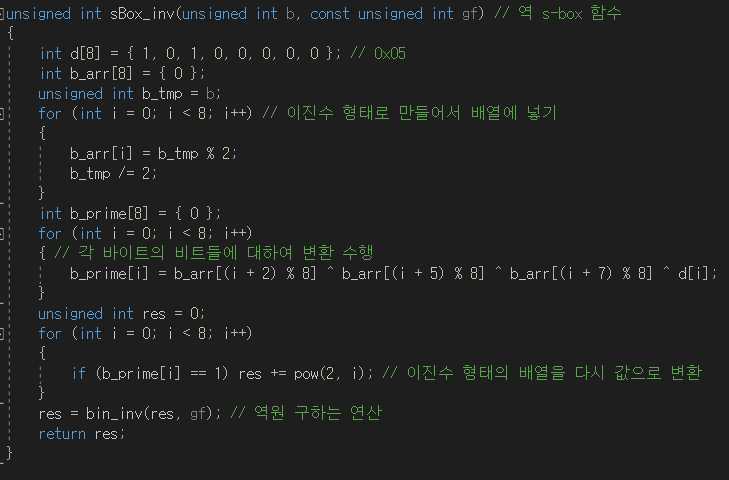
* 표준 입력으로 ‘d’를 입력 받았을 때는 decryption을 수행하고, ‘e’나 ‘d’가 아닌 다른 명령어를 입력 받았을 때 에러 메시지를 띄우는 코드입니다.
* Decryption 과정에서는 key가 w(40,43) ~ w(0,4) 순서로 적용되고, cipher.bin 파일을 입력 받고, round의 순서가 [inverse shift row – inverse substitution bytes – add round key – inverse mix column] 이 순서대로 진행되도록 encryption 과정과 차이를 두어 구현했습니다.
  + Key Expansion



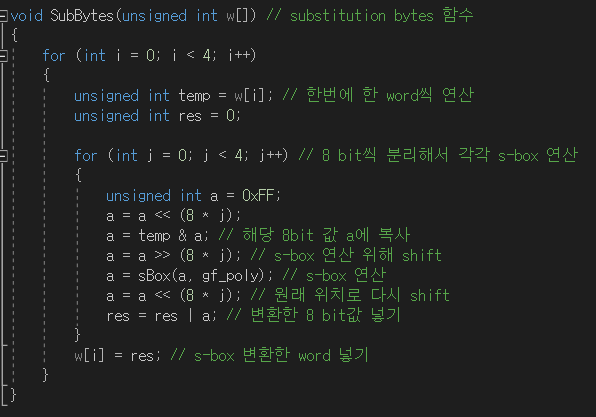
* Key expansion 함수입니다.
* 맨 처음 파일에서 입력 받은 key와 확장된 key가 저장될 배열을 인자로 받도록 구현하였습니다.

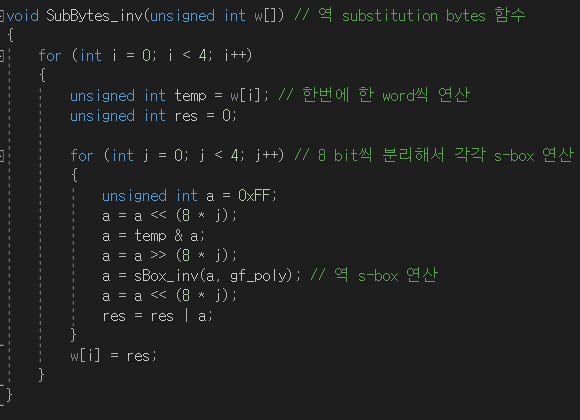
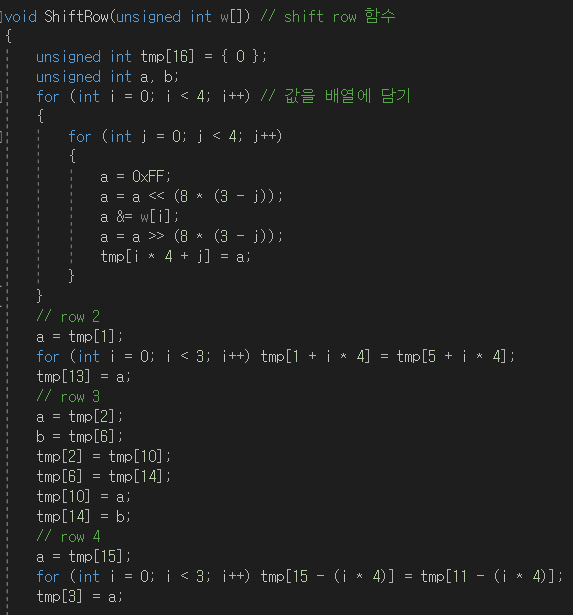


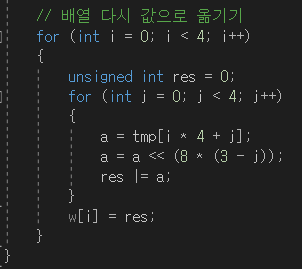
* 하나의 word와 round 수를 인자로 받는 g 함수입니다.
* 위치 이동과 각각 s-box 연산을 하는 코드입니다.
* Round 값에 따라 RC 값을 계산하고 shift 해준 후 XOR 연산을 하는 코드입니다.
* RC 값은 round(1~8), 9, 10에 따라 방식을 나눠서 계산하였습니다.
  + S-box
* S-box 연산을 하는 함수입니다.
* 인자로 받은 b값의 역원 구하는 연산을 해준 후, 이진수 형태의 배열에 넣은 후 비트들에 대하여 변환을 수행하고 다시 값을 변환하는 코드입니다.



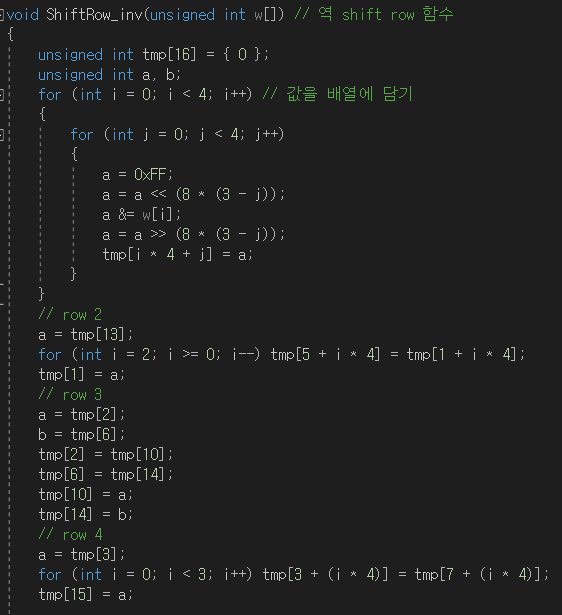
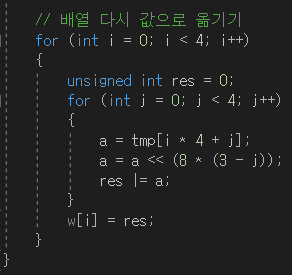
* 역 S-box 연산을 하는 함수입니다.
* 비트들에 대하여 위의 연산을 수행하도록 구현하였습니다.
  + Substitution bytes

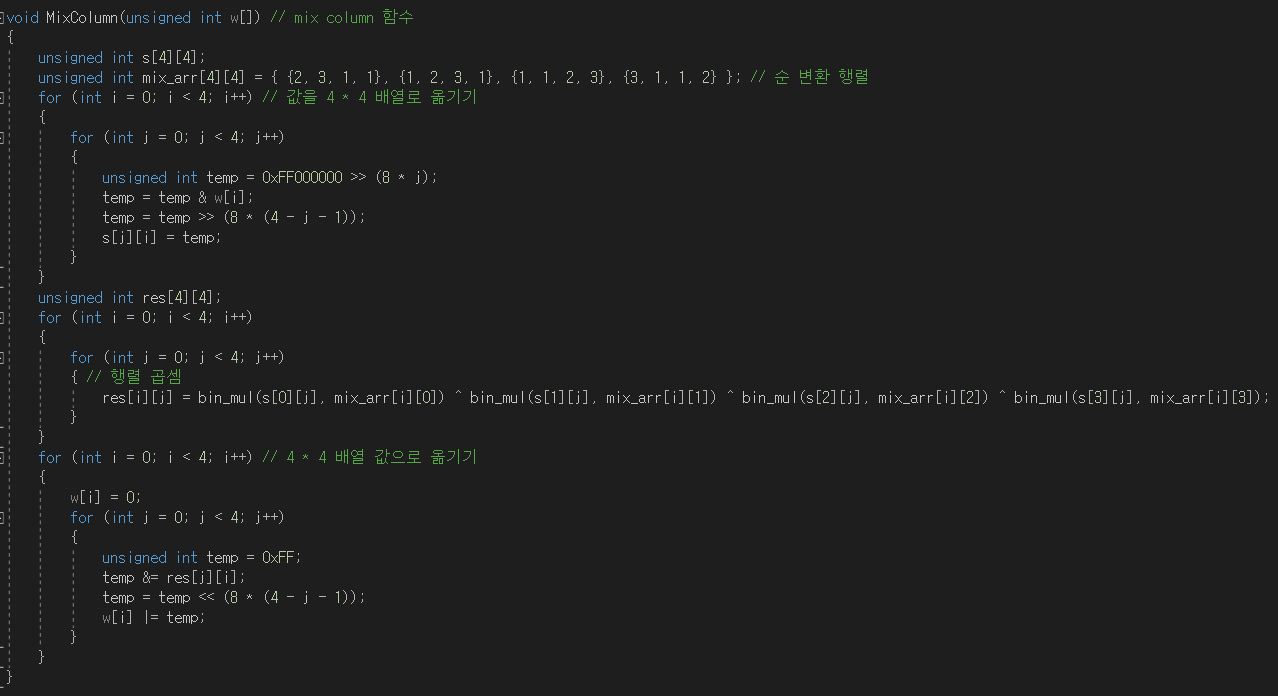
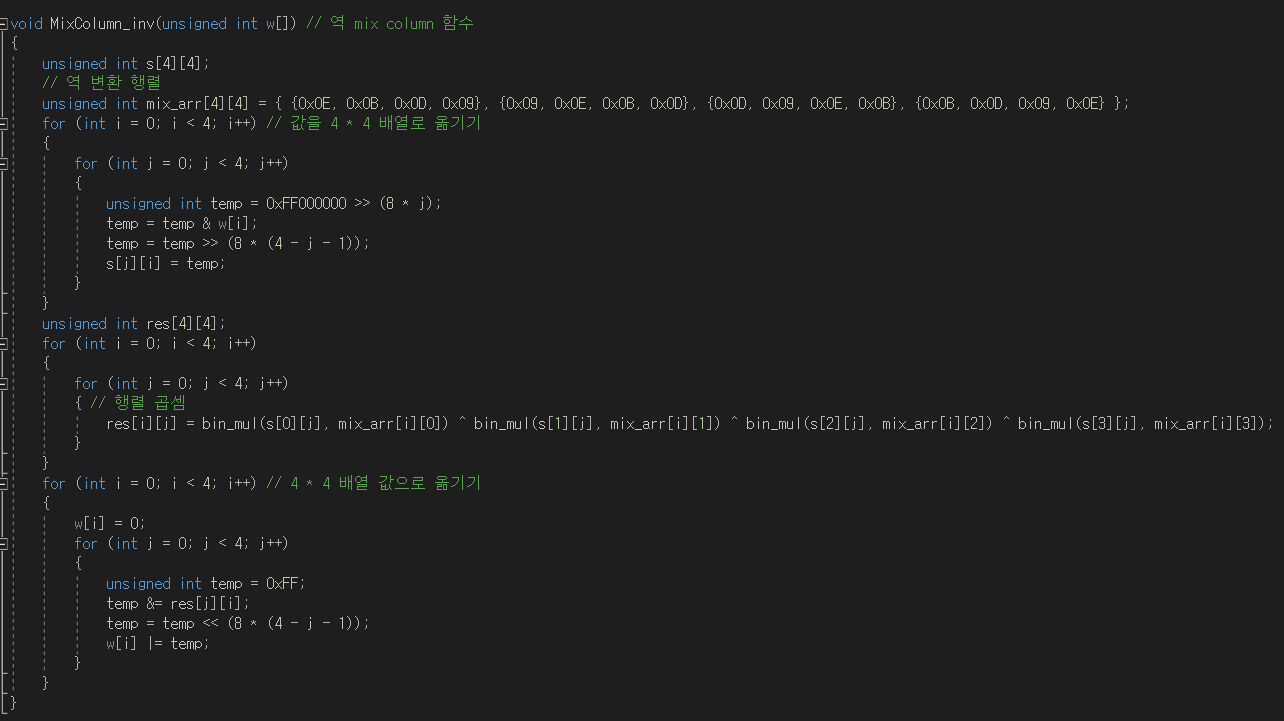


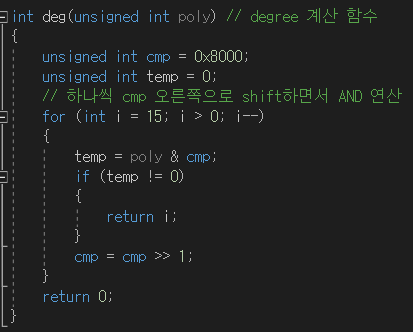
* Substitution bytes 연산을 하는 함수입니다.
* 상태 배열을 인자로 받아, 한번에 한 word씩, 한 word마다 8bit씩 분리해서 각각 s-box 연산을 하도록 구현하였습니다.
* 역 substitution bytes 연산은 다음과 같습니다.
  + Shift row



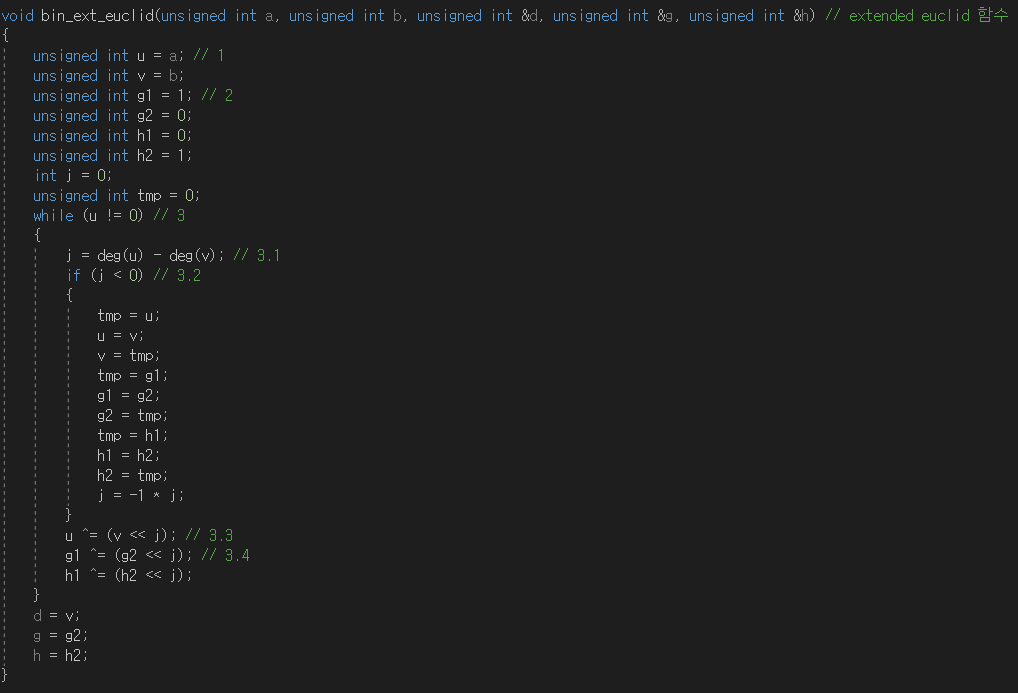
* Shift row 연산을 하는 함수입니다.
* 상태 배열을 인자로 받아 배열에 넣은 후, 2행은 1byte, 3행은 2byte, 4행은 3byte씩 왼쪽 순환 이동시켰습니다.



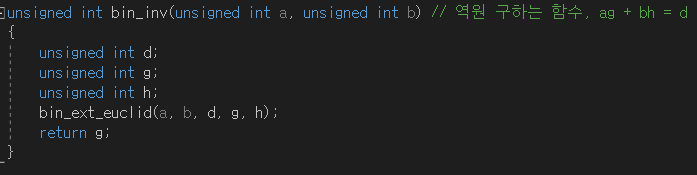
* 역 shift row 연산 함수입니다.
* 순환 이동하는 방향이 순방향과 반대인 오른쪽이 되도록 구현하였습니다.
  + Mix column
* Mix column 연산을 하는 함수입니다.
* 상태 배열을 인자로 받아 4 X 4 배열에 넣은 후 행렬 곱셈(순 변환 행렬과)을 한 뒤 다시 원래 값으로 옮기는 코드입니다.
* 역 mix column 연산을 하는 함수입니다.
* 행렬 곱셈에 쓰이는 행렬이 역 변환 행렬입니다.
  + Other functions



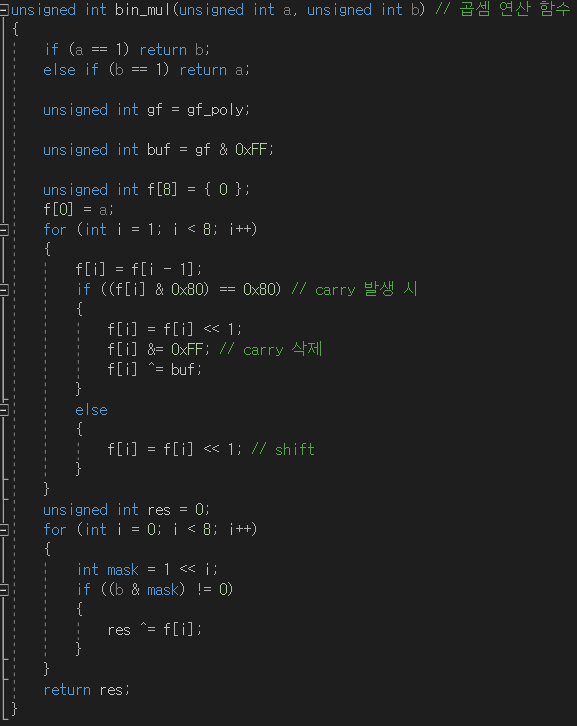
* Degree를 계산하는 함수입니다.



* Extended Euclid algorithm를 계산하는 함수입니다.



* 곱셈의 역원을 계산하는 함수입니다.



* 곱셈 연산 함수입니다.