정보보호

소프트웨어학과

2014041057 최동민

조오현 교수님

목차

[1. 블록 암호 3](#_Toc20586838)

[1) 블록 암호란? 3](#_Toc20586839)

[2) Feistel vs SPN 3](#_Toc20586840)

[3) 패딩 3](#_Toc20586841)

[4) ECB, CBC 모드 3](#_Toc20586842)

[2. AES 3](#_Toc20586843)

[3. Encrypt and Decrypt with AES 4](#_Toc20586844)

[1) Encrypt 4](#_Toc20586845)

[2) Decrypt 4](#_Toc20586846)

[4. Cracking AES with Brute-forcing 5](#_Toc20586847)

[5. Reference 7](#_Toc20586848)

# 블록 암호

## 블록 암호란?

블록 암호는 평문을 블록 단위로 암호화 하는 대칭 키 암호 시스템이다. 대칭 키 암호 시스템은 암호화와 복호화 시 같은 키를 사용하는 암호 방식이다. 크게 Feistel 구조와 SPN 구조로 나뉜다.

## Feistel vs SPN

Feistel은 암호화 과정에서 역함수가 필요 없지만, 라운드 함수를 안전하게 설계해야 한다는 단점이 있다. 대표적인 알고리즘으로는 DES가 있다. SPN은 역함수가 필요하도록 설계되어야 한다는 단점이 있지만, 중간에 비트의 이동 없이 한번에 암호화/복호화가 가능하기 때문에 Feistel 구조보다 효율적으로 설계가 가능하다.

## 패딩

블록 암호화 방법으로 평문을 암호화 하기 위해선 우선, 평문을 블록 크기의 배수로 만들어야 한다. 평문 길이가 블록 크기의 배수가 안 되면, 블록 크기의 배수가 되게끔 패딩을 추가한다. 패딩을 단순히 0000과 같이 추가할 경우 원문과 패딩을 구별하기 어려우므로, PKCS#5, ANSIX923 등과 같은 패딩 알고리즘이 사용된다.

## ECB, CBC 모드

ECB 모드는 평문을 블록 크기대로 잘라 독립적으로 암호화 하는 방식이다. 그러나 키가 고정일 경우 평문 내에서 동일한 블록은 동일한 암호문으로 암호화 되는 특징이 있어 보안에 취약하다. 이를 보완한 CBC모드 CTR 모드 등이 있다.

CBC 모드는 현재 블록을 암호화 할 때, 이전 블록의 암호화 결과를 XOR 한 후 암호화를 한다. 즉, 여러 평문 블록을 동시에 병렬적으로 암호화 할 수는 없다. n 번째 블록을 암호화 하기 위해서는 n-1 번째 블록을 암호화 한 결과가 필요하다.

# AES

AES는 Advanced Encryption Standard의 약어이며, DES의 짧은 키를 보완한 암호 알고리즘이다. 암호화, 복호화 과정에서 같은 키를 사용하는 대칭 키 알고리즘이다. NIST에서 표준으로 선정한 암호 알고리즘이며, WPA2에도 AES가 사용된다고 알려져있다. SPN 구조를 가지며, 중간에 키를 XOR하며 여러 라운드에 걸쳐 적용한다. 블록 크기는 통상 16 바이트(128 비트)이며, 4 X 4인 2차원 행렬 만들어 Substitution과 Permutation을 수행한다. 암호화의 한 라운드는 SubBytes, ShiftRows, MixColumns, AddRoundKey로 이루어져 있으며, 마지막 라운드에서는 MixColumns을 시행하지 않는다.

# Encrypt and Decrypt with AES

## Encrypt

AES 알고리즘으로 평문을 암호화 한다. AES 암/복호화 코드는 온라인 상[[1]](#footnote-1)에 공개 되어있는 코드를 사용했으며, 깃허브[[2]](#footnote-2)에 업로드 해놓았다. 아래 [그림 1]는 암호화 키 ABCDABCDABCDABCDAB CDABCD12345678를 이용해 평문 JungBoBoHo Gazua를 암호화 한 모습이다. 해당 평문의 hex 인코딩 값은 4a 75 6e 67 42 6f 42 6f 48 6f 20 47 61 7a 75 61이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

그림 1 - AES 암호화 결과

각 라운드 별 결과를 확인할 수 있으며, 최종 암호화 결과는 26 26 1d 7c 5f 83 09 a7 95 9e 00 97 05 60 a8 d3인 모습을 확인할 수 있다.

## Decrypt

AES는 대칭 키 암호 알고리즘이므로 암호화 할 때 사용한 키로 복호화 할 수 있다. 암호화 할 때 사용한 키와 암호문을 입력하면 아래 [그림 2]와 같이 평문을 얻을 수 있다.

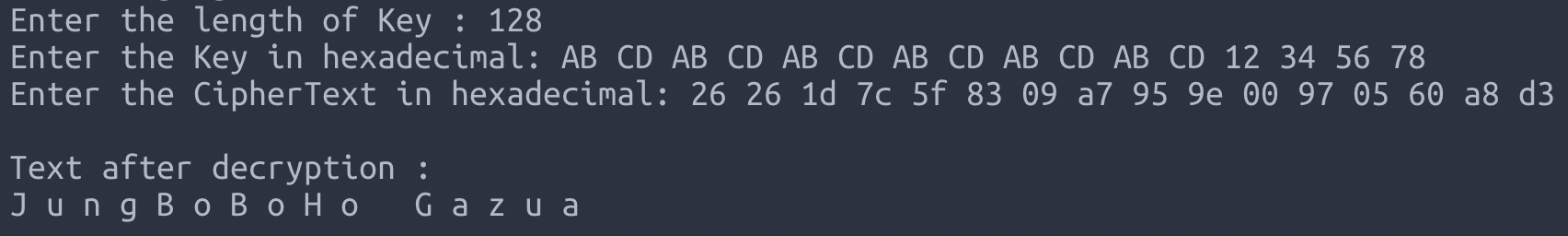


그림 2 - AES 복호화 결과

# Cracking AES with Brute-forcing

128비트의 AES키를 브루트 포스하여 맞추는 확률은 2128인 3402823669209384634633746074317 68211456가지 경우의 수 중 한 가지이다. 브루트 포싱 하는데 걸리는 시간을 예측하기 위해 224인 16777216번 만큼 걸리는 시간을 측정한 후, 그 2128/224인 2104만큼 곱해준다. 그러면 총 128비트 브루트 포싱 하는 시간을 예측할 수 있다. 그런데 이때, 한 번에 키를 찾을 확률이 1/2128이므로 평균적으로 2127번 만에 찾을 수 있다. 따라서 실제 계산 시에는 2128을 2로 나눈 2127 즉, 2127/224을 계산하면 된다. 224만큼 걸리는 시간 측정은 다음 [표 1]와 같이 진행한다.

|  |
| --- |
| t1 = time(NULL);  for(i=0; i<=0xffffff; i++){  encrypt(“평문”, “키”);  }  t2 = time(NULL); |

표 1 - 브루트 포싱 수도코드

t2-t1의 값이 24비트 브루트 포싱시 걸리는 시간이 된다. 그러나 이를 코드로 옮길 때 주의할 점이 있다. 반복문 수행 시마다 평문을 다음 암호문으로 계속 업데이트 해 주어야 한다는 것이다. 예를 들어, encrypt(0x1234, 0x5678)의 결과가 0xdeadbeef라면, 다음 암호화 수행은 encrypt(0xdeadbeef, 0x5678)와 같이해야 한다. 업데이트를 하지 않으면, 문자열을 메모리에 고정 해놓고 매번 계산하지 않고 바로 읽어오도록 되어있기 때문이다. 이는 컴파일러 최적화 문제 때문이며, 문자열이 메모리에 고정되어 있다면 encrypt() 함수 수행 시 매번 계산하지 않으므로 시간 측정이 제대로 되지 않는다. 따라서 입력 문자열을 매번 업데이트 해야 한다.

|  |
| --- |
| t1 = time(NULL);  for(i=0; i<=0xffffff; i++){  encrypt(“평문”, “키”);  memcpy(in, out, 16);  }  t2 = time(NULL); |

표 2 - 수정된 브루트 포싱 수도코드

아래 [그림 3]는 24비트 AES 브루트 포싱에 65초가 소요된 모습이다. 실행 환경은 i5-7360U (2.3 GHz Dual Core Intel Core i5) 노트북이며, gcc 최적화 옵션은 O0을 사용했다.

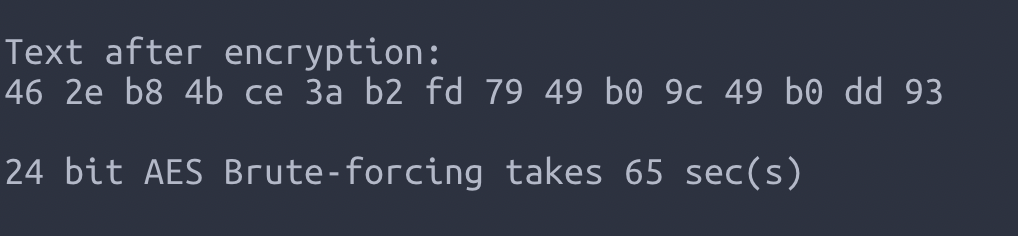


그림 3 - 노트북에서 소요 시간

2127/224은 10141204801825835211973625643008이고, 걸린 시간인 65초에 2127/224만큼 곱하면 총 AES 부르트 포싱 하는데 걸리는 시간을 계산할 수 있다. 총 6591783121186792887782856667955 20초가 걸릴 것이라고 예상할 수 있다. 환산하면 약 20902407157492367097231280년 정도이다.

다음은 i7-4770K (3.5Ghz Quad Core Intel Core i7) 데스크탑에서 측정한 모습이다. 윈도우즈 VS2017 환경이며 Release 모드 빌드하였고, 최적화 모드는 /Od를 사용하였다. 총 274초가 소요된 모습이다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

그림 4 - 데스크톱에서 소요 시간

다음은 AWS ec2 p2.xlarge에서 측정한 모습이다. Xeon(R) E5-2686 v4 (2.3Ghz 18 Core Intel Xeon(R)) CPU를 사용했으며, 월 100만원 이상의 고사양, 고성능 딥러닝 용 AWS 서버이다. 같은 gcc 최적화 옵션인 O0을 사용하였으며, 24비트 브루트 포스 결과는 아래 [그림 5]와 같이 116초가 걸렸다.

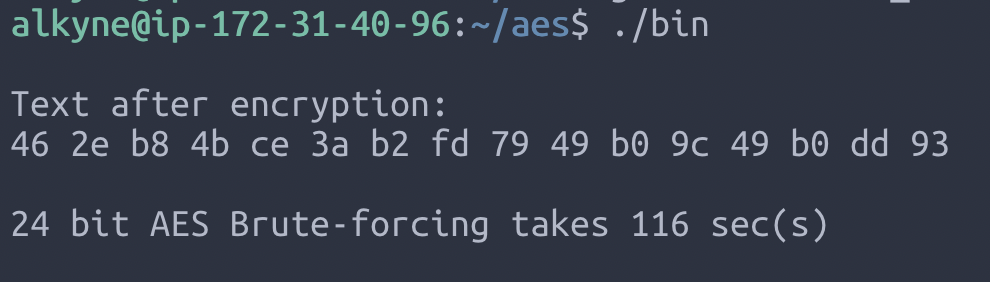


그림 5 - AWS 서버에서 소요 시간

고가의 딥러닝용 AWS서버 보다 노트북에서 더 좋은 성능을 낸 것이 다소 눈에 띈다. 여러 번 수행했지만 모두 같은 결과를 냈다. 벤치마크상[[3]](#footnote-3)에서 전체 CPU 성능은 서버에 비해 밀리지만, 싱글 스레드 부문에서 더 좋은 성능을 내서 이런 결과를 낳은 것으로 추정된다. 아래 [그림 6]는 벤치마크로 본 세 CPU의 비교이다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

그림 6 - 세 CPU 성능 비교

아래 [표 3]는 테스트 환경과 결과를 정리, 비교한 것이다.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Intel Core i5-7360U | Intel Core i7-4770K | Intel Xeon E5-2686 v4 |
| 용도(환경) | 노트북 | 데스크톱 | 서버 |
| OS | macOS 10.14.6 | Windows 10 Education | Ubuntu 16.04 LTS |
| 컴파일러 | gcc | gcc | VisualStudio 2017 (v141) |
| 최적화 | 사용 안 함 (O0) | 사용 안 함 (O0) | 사용 안 함 (/Od) |
| 24 비트 브루트 포스 소요 시간 | 65초 | 274초 | 116초 |
| 128 비트 브루트 포스 소요 시간 예상 (년) | 2090240715749  2367097231280 | 8811168555619  8593609867244 | 3730275738875  5608973520439 |

표 3 - 테스트 환경 및 결과 비교

# Reference

[AES 암/복호화 알고리즘 - 감파고](https://newstein03.tistory.com/1)

[안전한 패스워드 저장 - Naver D2](https://d2.naver.com/helloworld/318732)

[AES 암호 알고리즘 - Crocus](https://www.crocus.co.kr/1230)

[고급 암호화 표준 - Wekipedia](https://ko.wikipedia.org/wiki/고급_암호화_표준)

[암호학 - Block Cipher - Secreta](https://reinliebe.tistory.com/76)

[현대 암호1 : 블록암호 - blissotoner](http://www.secmem.org/blog/2019/02/06/block-cipher/)

[AES Implementation in C/C++ with comments - Hoozi Resources](http://www.hoozi.com/posts/advanced-encryption-standard-aes-implementation-in-cc-with-comments-part-1-encryption/)

1. http://www.hoozi.com/posts/advanced-encryption-standard-aes-implementation-in-cc-with-comments-part-1-encryption/ [↑](#footnote-ref-1)
2. github.com [↑](#footnote-ref-2)
3. https://www.cpubenchmark.net/compare/Intel-Xeon-E5-2686-v4-vs-Intel-i5-7360U/2870vs3036 [↑](#footnote-ref-3)