Rijndael 방식을 이용한 데이터 암호화 설계 및 구현

고훈*, 장의진**, 김대원***, 신용태***
*대진대학교 컴퓨터학과, **(주)디지캡, ***숭실대학교 컴퓨터학과

A design and implementation of data encryption using Rijndael Method

Hoon Ko*, Uj-jin Jang**, DaeWon-Kim***, Yong-Tae Shin***
*Department of Computer Science Daejin Univ. **Digicaps
****Department of Computer Science Soongsil Univ.

요 약

인터넷과 같은 네트워크와 컴퓨터 이용자의 급격한 증가로 인해 컨텐츠 유출 사고 및 개인 정보 유출 사고가 빈번하게 발생되고 있다. 이러한 문제를 해결하기 위한 방법 중의 하나가 컨텐츠를 암호화 하는 방법이다. 그러나 이 또한 최근에 암호화의 안전성에 많은 문제점을 안고 있다. 이에 본 논문에서는 비밀키 암호화 방식의 대표인 DES의 문제점에 대해서 분석하고 차세대 암호표준으로 선정한 대칭형 암호알고리즘인 Rijndael을 이용해서 파일의 암호화를 구현하고자 한다.

I. 서론

인터넷과 같은 네트워크와 컴퓨터 이용자의 급격한 증가로 인해 컨텐츠 유출 사고 및 개인 정보 유출 사고가 빈번하게 발생되고 있다. 이러한 문제를 해결하기 위한 방법 중의 하나가 컨텐츠를 암호화 하는 방법이다. 그러나 이 또한 최근에 암호화의 안전성에 많은 문제점을 안고 있다. 이에 본 논문에서는 암호화의 문제점에 대해서 분석하고 차세대 암호표준으로 선정한 대칭형 암호알고리즘인 Rijndael을 이용해서 파일의 암호화를 구현하고자 한다. AES(Advanced Encryption Standard)는 DES의 보안성에 문제점이 제기되어 이를 보완하고자 미국의 표준 기술연구소(NIST)에서 차세대 암호표준으로 선정한 대칭형 알고리즘이다. 본 논문의 구성은 다음과 같다. Ⅱ장에서는 비밀키 방식의 암호화에 대해서 설명하고 Ⅲ장에서는 Rijndael 방식에 대해서 설명하고 Ⅳ장에서는 Rijndael을 이용한 설계와 결과를 보여주고 Ⅴ장은 결과를 맺고자 한다.

II. 비밀키 방식의 암호화

비밀키 블록암호에서는 비밀키를 통상 어느 일정기간 고정해 사용하기 때문에 키 레이트가 적어지고 완전 비익을 달성할 수는 없다. 암호화 알고리즘을 암호의 설계자 이외에 언제까지나 비밀로 보존하는 것은 일반적으로 곤란하며, 또 암호의 사용자는 암호의 설계자로부터도 정보를 비밀로 보전하고 싶다고 하는 요구가 있다. 암호의 공격법은 공격자가 이용할 수 있는 평문 및 암호문의 종류에 의해 다음의 4종류로 분류할 수 있다.

(1) 암호문공격(cihpertest only attack) : 암호문만을 이용할 수 있는 경우

(2) 기지평문 공격(known plaintext attack) : 평문과 대응하는 암호문을 이용할 수 있는 경우

(3) 선택평문공격(chosen plaintext attack) : 공격자가 임의로 선택한 평문과 그것에 대응한 암호문을 이용할 수 있는 경우

(4) 선택암호문공격(chosen ciphertext attack) : 공격자가 임의로 선택한 암호문과 그것에 대응한 암호문을 이용할 수 있는 경우

DES란 Data Encryption Standard의 약자로 1977년 미국정부에 의해서 표준으로 정해진 블럭 암호화 알고리즘이다. 원래 IBM에서 개발되었고 전 세계적으로 가장 많이 알려지고 사용되는 암호 알고리즘이다. DES는 대칭키 암호화 알고리즘이다. 통신에 사용될 때 송신자와 수신자는 암화화와 복호화 할 때 사용되는 같은 비밀키를 알고 있어서 한다. DES는 사용자가 하드 디스크에 데이터를 암호화해서 보관하는 데에도 사용된다. 다중 사용자 환경에서 비밀키 분배가 어려울 수 있다.

이러한 문제점은 공개키 알고리즘에 의해서 해결되었다. DES는 64bit의 블럭 사이즈를 가지고 54bit의 키를 사용하여 암호화 한다. DES는 16라운드의 단계를 거치며 기본적으로 하드웨어제작을 기반으로 디자인 되었다. 따라서 소프트웨어로 제작된 DES보다 하드웨어로 제작된 DES의 수행속도가 훨씬 빠르다.

안전성

DES 알고리즘의 안전성에 대한 비판 중에서 기술적인 문제점은 다음의 세 가지가 이다.

(1) 환자(S박스)을 설계 기준이 공표되지 않았다.

(2) DES의 키의 길이(56비트)가 적절하지 않다.

(3) DES의 단수(16단)가 적절하지 않다.

III. Rijndael 방식

AES 암호 알고리즘은 몇몇의 조작은 byte level로 되어있다. 한정된 필드 $GF(2^8)$안의 요소를 byte 로 표시한다. 다른 조작은 4 byte words로 정의된다. 이 절에서 우리는 기본적인 수학적 개념을 필요로 하므로 여기에서 소개 한다. 논문에서는 DES 방식에 비해 보안성과 처리속도 메모리 요구량 등에서 우수하게 평가 받고 있는 Rijndael을 이용해서 구현하였다. Rijndael 방식은 timing attacks, power attacks, differential power analysis attacks에 대해서 DES 방식보다 우수하며, 처리 속도도 빠르다. 메모리 요구량도 RAM과 ROM 메모리의 매우 제한된 자원을 사용하여 제한된 자원만이 허락되는 환경에서도 사용할 수 있다.

1. The Field $GF(2^8)$

AES일반적으로, 유한필드 $GF(p^m)$는 p^m개의 원소들을 포함하며, 여기서 p는 소수이고, m은 양의 정수로 나타낸다. $GF(2^8)$는 { 00000000 ~ 11111111 }으로 구성된 2진 비트 수열은 계수가 {0,1} 인 다항식으로 표현할 수 있다. AES에서는 유한 필드 $GF(2^8)$을 사용하며 2^8개의 원소들을 포함한다. 이진 다항식으로 아래와 같이 표현될 수 있다. 바이트 b는 b7 b6 b5 b4 b3 b2 b1 b0의 8 비트로 되어있다. 이 다항식은 0과 1의 계수로 구성된다.

$$b_7 x^7 + b_6 x^6 + b_5 x^5 + b_4 x^4 + b_3 x^3 + b_2 x^2 + b_1 x^1 + b_0$$

2. Addition

A다항식의 표현에서 두 요소의 합은 두 항의 요소의 2진 합에 의해서 주어지는 다항식의 요소이다. 덧셈은 XOR 연산으로 이행 된다 즉, mod 2에 대한 비트 단위의 덧셈이라 할 수 있다.

3. Multiplication

다항식의 표현에서, 더 이상 줄일 수 없는 8차 2진 다항식 모듈의 곱셈과 $GF(2^8)$의 곱셈은 일치한다. 다항식이 만약 더 이상 줄일 수 없다면 그것은 약수에 지나지 않는 1과 그 자신이다. AES에서 이 다항식은 16진수로 표현된

$m(x) = x^8 + x^4 + x^3 + x + 1$에 의해 주어진 $m(x)$를 호출한다.

4. Polynomials with Coefficients in $GF(2^8)$

다항식들은 일치하는 계수가 간단한 덧셈에 의해서 더해진다. GF의 덧셈은 비트 단위의 XOR이다.

$$a(x) = a_3 x^3 + a_2 x^2 + a_1 x + a_0$$

$$b(x) = b_3 x^3 + {}_2 x^2 + b_1 x + b_0$$

위 식의 덧셈을 아래와 같이 표현될 수 있다.

$$c(x) = c_6 x^6 + c_5 x^5 + c_4 c^4 + c_3 x^c + c_2 c^2 + c_1 x + c_0$$

IV. 설계 및 구현

AES 암호 알고리즘은 입출력이 블록 단위로 동작하는 알고리즘이다. 기본적인 단위는 Byte 단위로 작동하며, 내부적으로는 2차원 배열 형태의 State 단위로 동작한다. 여기서는 128bit 키, 블록과 10라운드를 기준으로 구현하였다. 각 라운드 변환은 외부에서 주어진 1차원 형태의 128비트 블

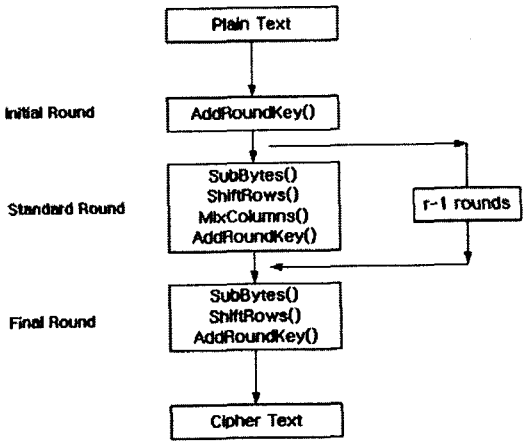


그림 3 : 암호화 과정

록을 2차원 4행× N_b(단, N_b=4구성)로 구성되는 State로 변환한 후, State내의 배열에 대해서 연산을 수행한다.

1. 암호화

AES의 전반적인 암호화 과정은 [그림 1]에 잘 나타나 있다. 평문이 들어오면 먼저 AddRoundKey() 가 동작되고, r-1라운드까지는 4개의 변환으로 구성되어 있다. SubBytes(), ShiftRows(), MixColumns(), AddroundKey()으로 구성되어 있으며, 마지막 라운드는 3개의 변환 있는데 SubBytes(), ShiftRows(), AddRoundKey()로 구성되어 있다.

```
Rijndael(byte State, byte CipherKey)
{
  KeyExapansion(CipherKey, ExpandedKey);
  AddRoundKey(State, ExpandedKey);
  for(i=1;i<Nr;i++)
    Round(State, ExpandedKey+Nb*i);
  FinalRound(State, ExpandedKey+Nb*Nr);
}
```

2. 복호화

AES의 복호화 과정은 다른 블록 암호 알고리즘과 다르게 암호화 과정이 다르다. 암호문이 들어 오면 먼저 AddRoundKey()가 동작되고, 암호화 과정과 마찬가지로 r-1 라운드까지는 4개의 변환으로 구성되어 있다.

InvShiftRows(),InvSubBytes(),AddroundKey(), InvMixColumns()으로 구성되어 있으며, 마지막 라운드는 3개의 변환 있는데 InvShiftRows(), InvSubBytes(), AddRoundKey()로 구성되어 있다.

```
InvRijddael(byte State, byte CipherKey)
{
  InvKeyExpansion(CipherKey, InvExpandedKey);
  AddRoundKey(State, InvExpandedKey+Nb*Nr);
  for(i=Nr-1;i>0;i--)
      InvRound(State, InvExpanedKey+Nb*i);
  InvFinalRound(State, InvExpandedKey);
}
```

1) SubBytes() / InvSubBytes()

SubBytes() 변환은 S-Box를 사용한 각각의 State byte들이 독립적으로 비선형 바이트 치환을 한다. S-Box는 두 가지 변환에 의해서 만들어진다. $GF(2^8)$에서의 곱셈상의 역을 구하거나 $GF(2^8)$에 대해 아래의 연산을 수행한다. 연산을 수행하면 $2^8 \times 8$ 의 룩업 테이블 형태의 S-Box로 구현된다.

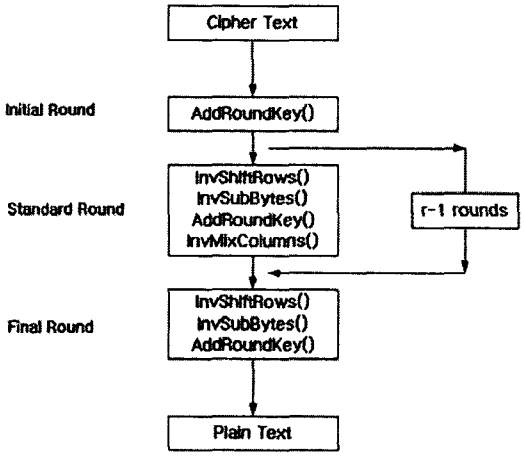


그림 4 : 복호화 과정

InvSubBytes() 변환은 바이트 치환 변환의 역변환이다. 역 S-Box가 각각의 State에 적용되어진다.

```
alog[0] = 1;
    for (i = 1; i < 256; i++) {
        j = (alog[i-1] << 1) ^ alog[i-1];
        if ((j & 0x100) != 0) j ^= ROOT;
        alog[i] = j;
}
byte[][] A = new byte[][] {
        :
};
byte[] B = new byte[] { 0, 1, 1, 0, 0, 0, 1, 1};
```

2) ShiftRows() / InvShiftRows()

ShiftRows() 변환은 State의 마지막 3행의 바이트들을 순환적으로 서로 다른 수의 바이트를 쉬프트 시킨다.

InvShiftRows() 변환은 ShiftRows()의 역변환이다.

```
for (i = 0; i < 256; i++) {
        S[i] = (byte)(cox[i][0] << 7);
        for (t = 1; t < 8; t++)
            S[i] ^= cox[i][t] << (7-t);
        Si[S[i] & 0xFF] = (byte) i;
}
```

3) MixColumns() / InvMixColumns()

MixColumns()변환 State의 column-by-column 연산하는 것이다. 각각의 열은 $GF(2^8)$에서의 다항식으로 표현되어진다. 고정된 식 $a(x)$를 모듈 x^4+1로 Multiplication을 한다.

InvMixColumns() 변환은 MixColumn()의 역변환이며, column-by-column 연산하는 것이다. 각각의 열은 $GF(2^8)$에서의 다항식으로 표현되어진다.

```
static final int multiplication (int a, int b) {
    return (a != 0 && b != 0) ?
    alog[(log[a & 0xFF] + log[b & 0xFF]) % 255] :
    0;
}
                :
// InvMixColumns()
 for (int r = 1; r < ROUNDS; r++)
     for (j = 0; j < BC; j++) {
     tt = Kd[r][j];
         Kd[r][j] = U1[(tt >>> 24) & 0xFF] ^
       U2[(tt >>> 16) & 0xFF] ^
       U3[(tt >>>  8) & 0xFF] ^
       U4[ tt         & 0xFF];
 }
```

4) AddRoundKey()

AddroundKey() 변환은 State값과 라운드 키 간의 XOR 연산으로 처리된다.

AddRoundKey(State, RoundKey);

5) Key Expansion

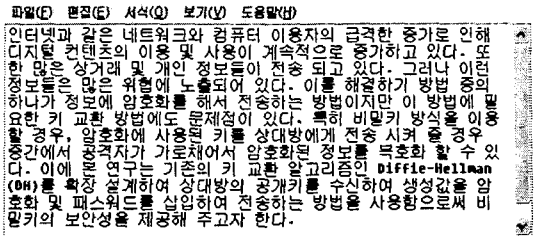
AES 알고리즘은 암호화 키 K를 가지는데 키 확장 루틴은 키 스케줄에 의해 만들어진다. 확장된 키는 4바이트의 선형배열 ($w[\ N_b(\ N_r+1)]$)로 표시된다. 초기화 N_b가 필요하며 각각의 라운드 N_r은 N_b 워드의 키 데이터가 요구되어진다.

KeyExpansion(byte CipherKey, word ExpandedKey);

Rijndael을 이용한 구현 결과는 아래와 같다.

V. 결론

본 논문은 비대칭 암호화 방식의 차세대 방식인 Rijndael 방식에 대해서 분석을 해봤고, Rijndael 방식을 이용해서 간단한 파일을 암호화 하였다. 앞에 밝혔듯이 Rijndael 방식은 DES 방식과 비교해 봤을 때, 속도가 빠르고, 작은 코드로 인해 메모리 요구량이 작아서 스마트카드 등에 사용이 용이하다.


파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

인터넷과 같은 네트워크와 컴퓨터 이용자의 급격한 증가로 인해 디지털 컨텐츠의 이용 및 사용이 계속적으로 증가하고 있다. 또한 많은 상거래 및 개인 정보들이 전송 되고 있다. 그러나 이런 정보들은 많은 위협에 노출되어 있다. 이를 해결하기 방법 중의 하나가 정보에 암호화를 해서 전송하는 방법이지만 이 방법에 필요한 키 교환 방법에도 문제점이 있다. 특히 비밀키 방식을 이용할 경우, 암호화에 사용된 키를 상대방에게 전송 시켜 줄 경우 중간에서 공격자가 가로채어서 암호화된 정보를 복호화 할 수 있다. 이에 본 연구는 기존의 키 교환 알고리즘인 Diffie-Hellman (DH)를 확장 설계하여 상대방의 공개키를 수신하여 생성값을 암호화 및 패스워드를 입력하여 전송하는 방법을 사용함으로써 비밀키의 보안성을 제공해 주고자 한다.

그림 5 : 암호화 전

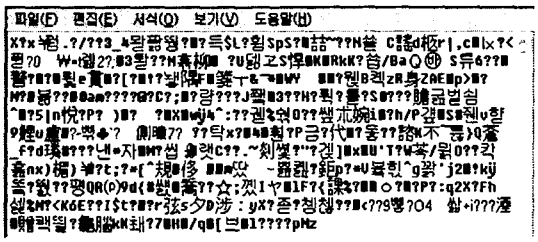

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

그림 6 : 암호화 후

그러나 어느 암호화 알고리즘이 마찬가지겠지만 암호화 후에 키를 전송하는 과정에서 많은 문제점이 있다. 본 암호화 방식도 키 전달 방법에 약점이 있다. 최근 많은 키 분배 방법에 대해서 제안되고 있지만 이 또한 완벽하지는 않다. 향후에는 본 알고리즘을 바탕으로 한 안전한 키 교환 방법에 관해서도 연구가 필요하다.

참고문헌

[1] D, E, Denning : "Cryptography and data security", ADDISON Wesley Pub, 1982

[2] IST, NIST SP 800-16, "*Information Technology Security Training Requirement*", 1998.

[3] S. Bellovin and M. Merritt. "Augmented Encrypted Key Exchange : A Password-Based Protocol Secure against Dictionary Attacks and Password File Compromise," ACM Conference on Computer and Communications Security 1993, pp. 244~250.

[4] 이원구, 이재광, "자바기반의 타원곡선 알고리즘을 이용한 보안 메일 시스템의 설계 및 구현," 한국정보보호 진흥원, 2001