2019 암호경진대회 : 4번 문제 답안

압축함수 H의 특징을 살펴보면,

1. 입력 Message 256비트를 반 씩 나눈다.

반복문에 의해 Xi+1 비트들은 Ki 비트의 영향을 받는다.

1. H의 결과값은 X 비트, 그리고 X0 , K0 즉 고정된 입력 Message의 XOR 연산이다.

압축함수 H의 연산 과정을 수식으로 나타내어 보았다.

먼저, X0 을 32비트씩 나눠 A,B,C,D라 하고 K0 을 32비트씩 나눠 E,Z,G,H라고 하자.

그리고 F(rk, x) = Frk(x)라고 표현하면

예를 들어 j=3일 때, H = X2 ⊕ X0 ⊕K0

= FE(A)⊕ B⊕ A⊕ E , FE’(FE(A)⊕ B)⊕ C⊕ B⊕ Z

D⊕ C⊕ G , A⊕ D⊕ H 이다.

J가 증가할수록 Xi에 일정한 패턴이 생긴다. ⊕ 연산 앞에 F함수에 의해 묶이는 부분과 ⊕ 연산 뒤에 나오는 부분으로 나눌 수 있다. 이 패턴을 이용하여 충돌쌍을 찾아보았다.

j=5일때의 A~D까지의 비트더미들의 식을 풀어써 보면,

**A'** = FE’’’(FE’’(FE’(FE(A)⊕ B)⊕ C)⊕ D)⊕ A ⊕ A ⊕ E

**B'** = FE’’’’(FE’’’(FE’’(FE’(FE(A)⊕ B)⊕ C)⊕ D)⊕ A) ⊕ FE(A)⊕ B ⊕ B ⊕ Z

**C'** = FE’(FE(A)⊕ B)⊕ C ⊕ C ⊕ G

**D'** = FE’’(FE’(FE(A)⊕ B)⊕ C)⊕ D ⊕ D ⊕ H

이러한 경우에 보게되면 A ⊕ A 같이 자기자신을 XOR하게 되면 0이되는 특성을 이용하여 FE(A)⊕ B만의 충돌쌍을 찾아서 브루트포스의 경우의수를 줄일 수 있다.

즉, 같은 FE(A)⊕ B 값을 가지게 되는 서로 다른 A와 B를 구하게 된다면, 같은 **B'**값또한 찾을 수 있다.

같은 FE(A)⊕ B 값을 가지게 되는 서로 다른 A와 B를 구할때, 우리는 B값을 고려하지 않아도 된다. 왜냐하면 B ⊕ B는 0이며 나머지 값들에 대해서는 B의 값이 영향을 미칠때 FE(A)⊕ B의 형태로만 존재하기 때문이다.

**A'** = FE’’’(FE’’(FE’(FE(A)⊕ B)⊕ C)⊕ D)⊕ A ⊕ A ⊕ E

**B'** = FE’’’’(FE’’’(FE’’(FE’(FE(A)⊕ B)⊕ C)⊕ D)⊕ **A**) ⊕ **FE(A)**⊕ B ⊕ B ⊕ Z

**C'** = FE’(FE(A)⊕ B)⊕ C ⊕ C ⊕ G

**D'** = FE’’(FE’(FE(A)⊕ B)⊕ C)⊕ D ⊕ D ⊕ H

입력값 A만 바꾼다고 할때, 고려해야 할 대상은 빨간 표시의 'A'와 'FE(A)' 뿐이다. 이는 곧 결과 값 **B'** 뿐이다. 입력값 32Bit(A)를 바꿨을때 결과값 32Bit(B')만 바뀐다는것은 비둘기집 원리에 의해 일대일 대응이 아닌이상 충돌이 일어날 수 밖에 없다. 충돌쌍을 찾는데에 쓰이는 A값의 경우의 수는 2^32를 가진다.

실제로 충돌쌍을 찾을 때에는 FE(A)⊕ B = 0 으로 정해놓은 뒤 A값만 바꾸도록 하였다. 그렇게 해서 충돌쌍을 일으키는 A에 대하여 FE(A)를 구하여 B값으로 입력해주면 충돌쌍을 만들 수 있다. (FE(A)⊕ B = 0 성립)

이렇게 하여 찾은 j=5 일때의 충돌쌍

메세지 1:

00 00 00 00 63 63 63 63 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

메세지 2:

27 00 00 00 26 CC CC 89 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

둘의 **해쉬값**

**[0x92, 0x17, 0xc2, 0x2d, 0x15, 0x1d, 0x 83, 0x f1, 0xe5, 0x02, 0x07, 0x03, 0xa5, 0x8a, 0x8d, 0x47]**

**<구현 코드>**

uint8\_t s\_box[16][16] = {

{0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5, 0x30, 0x01, 0x67, 0x2b, 0xfe, 0xd7, 0xab, 0x76},

{0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59, 0x47, 0xf0, 0xad, 0xd4, 0xa2, 0xaf, 0x9c, 0xa4, 0x72, 0xc0},

{0xb7, 0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7, 0xcc, 0x34, 0xa5, 0xe5, 0xf1, 0x71, 0xd8, 0x31, 0x15},

{0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05, 0x9a, 0x07, 0x12, 0x80, 0xe2, 0xeb, 0x27, 0xb2, 0x75},

{0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0, 0x52, 0x3b, 0xd6, 0xb3, 0x29, 0xe3, 0x2f, 0x84},

{0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b, 0x6a, 0xcb, 0xbe, 0x39, 0x4a, 0x4c, 0x58, 0xcf},

{0xd0, 0xef, 0xaa, 0xfb, 0x43, 0x4d, 0x33, 0x85, 0x45, 0xf9, 0x02, 0x7f, 0x50, 0x3c, 0x9f, 0xa8},

{0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38, 0xf5, 0xbc, 0xb6, 0xda, 0x21, 0x10, 0xff, 0xf3, 0xd2},

{0xcd, 0x0c, 0x13, 0xec, 0x5f, 0x97, 0x44, 0x17, 0xc4, 0xa7, 0x7e, 0x3d, 0x64, 0x5d, 0x19, 0x73},

{0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88, 0x46, 0xee, 0xb8, 0x14, 0xde, 0x5e, 0x0b, 0xdb},

{0xe0, 0x32, 0x3a, 0x0a, 0x49, 0x06, 0x24, 0x5c, 0xc2, 0xd3, 0xac, 0x62, 0x91, 0x95, 0xe4, 0x79},

{0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e, 0xa9, 0x6c, 0x56, 0xf4, 0xea, 0x65, 0x7a, 0xae, 0x08},

{0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6, 0xb4, 0xc6, 0xe8, 0xdd, 0x74, 0x1f, 0x4b, 0xbd, 0x8b, 0x8a},

{0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e, 0x61, 0x35, 0x57, 0xb9, 0x86, 0xc1, 0x1d, 0x9e},

{0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e, 0x94, 0x9b, 0x1e, 0x87, 0xe9, 0xce, 0x55, 0x28, 0xdf},

{0x8c, 0xa1, 0x89, 0x0d, 0xbf, 0xe6, 0x42, 0x68, 0x41, 0x99, 0x2d, 0x0f, 0xb0, 0x54, 0xbb, 0x16}

};

uint8\_t S(uint8\_t x) {

unsigned char t1 = x >> 4;

unsigned char t2 = x << 4;

t2 = t2 >> 4;

int x1 = (int)t1;

int x2 = (int)t2;

return s\_box[x1][x2];

}

uint8\_t vli\_mul(int a, uint8\_t xo) { // 8비트 유한체 곱

if (a == 1) {

return xo;

}

uint8\_t x;

if ((xo >> 7) == 0)

x = xo << 1;

else

x = (xo << 1) ^ 27;

if (a == 3) {

x = xo ^ x;

}

else if(a!=2)

printf("Error: a가 2와 3이아닌 잘못된 a \* x 유한체 곱입니다.\n");

return x;

}

void MixColumns(uint8\_t \*x) {

int i, j, k;

uint8\_t a[4][4] = {

{ 0x2, 0x3, 0x1, 0x1},

{ 0x1, 0x2, 0x3, 0x1},

{ 0x1, 0x1, 0x2, 0x3},

{ 0x3, 0x1, 0x1, 0x2 }};

uint8\_t tmp[4];

for (i = 0; i < 4; i++) {

uint8\_t temp[4] = { 0, };

for (j = 0; j < 4; j++) {

temp[j] = vli\_mul(a[i][j], x[j]);

}

tmp[i] = temp[0] ^ temp[1] ^ temp[2] ^ temp[3];

}

for (int i = 0; i < 4; i++)

x[i] = tmp[i];

}

void F(uint8\_t \*b, uint8\_t \*rk, uint8\_t \*x) {

// 이때 입력 rk와 x는 8비트짜리 4개(총 32비트 2개)

uint8\_t a[4];

for (int i = 0; i < 4; i++)

a[i]=x[i] ^ rk[i];

for (int i = 0; i < 4; i++)

b[i] = S(a[i]);

MixColumns(b);

return;

}

void Round(uint8\_t \*Y, uint8\_t \*rk, uint8\_t \*X) {

// 8비트 \* 4개 = rk, 8비트 \* 16개 = X

uint8\_t tmp\_F[4];

F(tmp\_F, rk, X);

uint8\_t tmp[16];

for (int i = 0; i < 4; i++) {

tmp[i] = tmp\_F[i] ^ X[4 + i];

tmp[i + 4] = X[8 + i];

tmp[i + 8] = X[12 + i];

tmp[i + 12] = X[i];

}

for (int i = 0; i < 16; i++)

Y[i] = tmp[i];

return 0;

}

void \_\_Round(uint8\_t \*Y, uint8\_t \*rk, uint8\_t \*X) {

// 8비트 \* 4개 = rk, 8비트 \* 16개 = X

uint8\_t tmp\_F[4];

F(tmp\_F, rk, X);

uint8\_t tmp[16];

for (int i = 0; i < 4; i++) {

tmp[i] = X[i];

tmp[i + 4] = tmp\_F[i] ^ X[4 + i];

tmp[i + 8] = X[8 + i];

tmp[i + 12] = X[12+i];

}

for (int i = 0; i < 16; i++)

Y[i] = tmp[i];

return 0;

}

void hash(uint8\_t \*H, int j,uint8\_t \*m) {

uint8\_t X[16], K[16],tmp[4],B[4];

uint8\_t E[4] = { 0x00, 0x00, 0x00, 0x00 };

for (int i = 0; i < 4; i++) {

B[i] = m[i + 4];

}

for (int i = 0; i < 16; i++) {

X[i] = m[i];

K[i] = m[i + 16];

}

uint8\_t zero\_X[16], zero\_K[16];

for (int i = 0; i < 16; i++) {

zero\_X[i] = X[i];

zero\_K[i] = K[i];

}

uint8\_t rk[4];

for (int i = 0; i < j - 1; i++) {

for (int k = 0; k < 4; k++)

rk[k] = K[k]; // rk = 32비트

Round(X, rk, X);

uint8\_t i\_tmp[4];

for (int k = 0; k < 4; k++)

i\_tmp[k] = i + k;

Round(K, i\_tmp, K);

if (i == 0) {

for (int k = 0; k < 4; k++)

X[k] = 0x00; // Fe(A) xor B = 0 으로 만드는 부분

}

}

for (int i = 0; i < 16; i++)

H[i] = K[i];

for (int k = 0; k < 4; k++)

rk[k] = K[k];

\_\_Round(X, rk, X);

for (int i = 0; i < 16; i++)

H[i] = X[i] ^ zero\_X[i] ^ zero\_K[i];

F(tmp, E, m);

for (int i = 4; i < 8; i++)

H[i] = H[i] ^ tmp[i-4] ^ B[i-4];

}

int compare(uint8\_t \*key, uint8\_t \*m) { // KEY 해쉬값과 같은지 비교하는 함수

int count = 0;

for (int i = 0; i < 16; i++) {

if (key[i] == m[i])

count++;

else

break;

}

if (count == 16)

return 1;

else

return 0;

}

int main() {

uint8\_t key[16];

printf("\n");

uint8\_t t[32]=

{ 0x00 , 0x01 , 0x02 , 0x03 , 0x04 , 0x05 , 0x06 , 0x07 ,

0x08 , 0x09 , 0x0a , 0x0b , 0x0c , 0x0d , 0x0e , 0x0f ,

0x0a , 0x1a , 0x2a , 0x3a , 0x4a , 0x5a , 0x6a , 0x7a ,

0x8a , 0x9a , 0xaa , 0xba , 0xca , 0xda , 0xea , 0xfa };

uint8\_t m[32] =

{ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };

uint8\_t test\_m[32] = //

{ 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 }; //

uint8\_t E[4] = { 0x00, 0x00, 0x00, 0x00 };

uint8\_t H[16];

uint8\_t tmp[4];

hash(H, 5, m);

for (int i = 0; i < 16; i++)

key[i]=H[i];

printf("\n");

long long i = 0;

printf("\n원본 메시지:\n ");

for (int k = 0; k < 32; k++)

printf("%02x ", m[k]);

printf("\n<<<<<<<<<<<<시작>>>>>>>>>>>\n");

printf("찾을키 :\n");

for (int i = 0; i < 16; i++) {

printf("%02x ", key[i]);

}

while(1) {

int carry = 0;

//

hash(H, 5, test\_m);

//

int res = compare(H, key);

if (res == 1) {

printf("\nfind!\n");

for (int k = 0; k < 16; k++)

printf("%02x ", H[k]);

printf("\n충돌 메시지:\n ");

for (int k = 0; k < 32; k++)

printf("%02x ", test\_m[k]);

printf("\n");

break;

}

do

{

test\_m[carry]++;

if (test\_m[carry] == 0x00) {

if (carry == 4){

printf("못찾");

return 0;

}

carry++;

continue;

}

if (carry > 2) {

printf("\n>>>>%d 돌파>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>\n", carry);

for (int i = 0; i < 32; i++) {

printf("%02x ", test\_m[i]);

}

printf("\n");

for (int i = 0; i < 16; i++) {

printf("%02x ", H[i]);

}

}

carry=0;

} while (carry != 0);

}

F(tmp, m, E);

printf("\n========================================\n");

printf("메세지1의 Fe(A)값 : ");

for (int i = 0; i < 4; i++) {

printf("%02x ", tmp[i]);

}

printf("\n");

F(tmp, test\_m, E);

printf("메세지2의 Fe(A)값 : ");

for (int i = 0; i < 4; i++) {

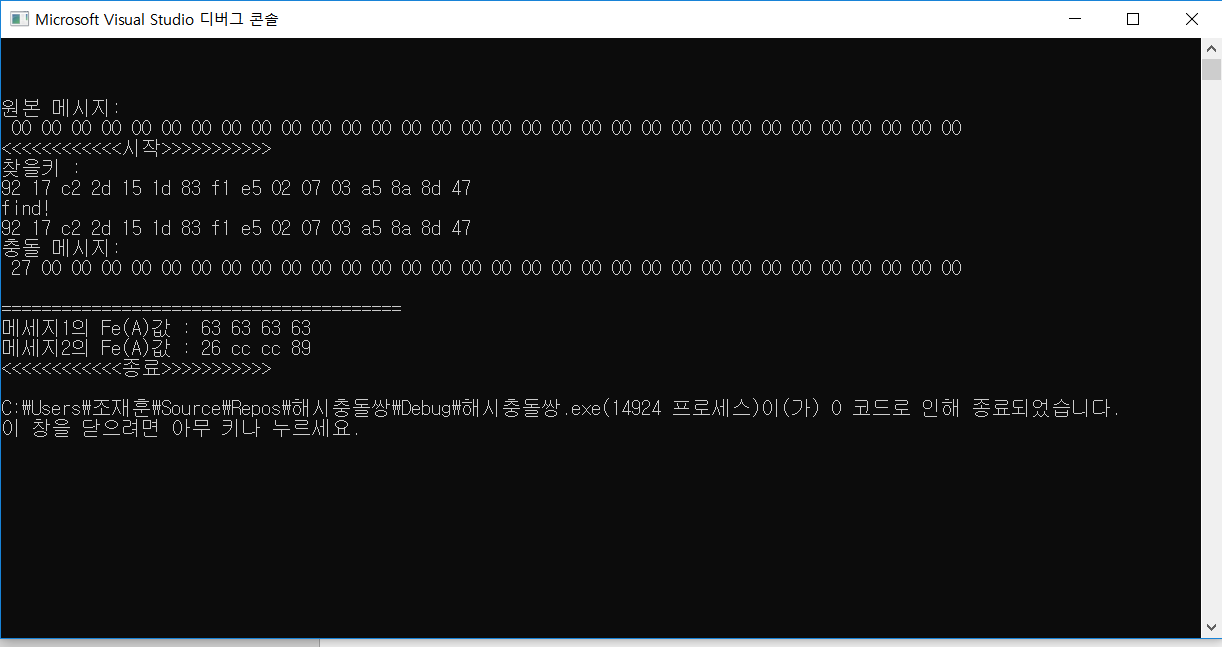
printf("%02x ", tmp[i]);

}

printf("\n<<<<<<<<<<<<종료>>>>>>>>>>>\n");

}

**<결과창>**

****

**<답>**

**Message :**

**00 00 00 00 63 63 63 63 00 00 00 00 00 00 00 00**

**00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00**

**Message' :**

**27 00 00 00 26 CC CC 89 00 00 00 00 00 00 00 00**

**00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00**

**Comp ( j , Message ) = Comp ( j , Message' ) :**

**{ 0x92, 0x17, 0xC2, 0x2D, 0x15, 0x1D, 0x83, 0xF1, 0xE5, 0x02, 0x07, 0x03, 0xA5, 0x8A, 0x8D, 0x47 }**