카운터를 사용한 블록암호 운영모드에 관한 연구

양상근* · 김길호* · 박창수* · 조경연*

*부경대학교 컴퓨터공학과

Study for Block Cipher Operating Mode Using Counter

Sang-Keun Yang* · Gil-Ho Kim* · Chang-Soo Park* · Gyeong-Yeon Cho*

*Dept of Computer Engineering Pukyong National Univ.

E-mail : apartee@naver.com

요 약

본 논문에서는 ASR(Arithmetic Shift Register)을 이용한 블록암호 운영모드를 제안한다. ASR이란 $GF(2^n)$상에서 0이 아닌 초기 값 A_0에 0 또는 1이 아닌 임의의 수 D를 곱하는 수열로서 산술시프트 레지스트라 부른다. 본 논문에서 제안하는 모드는 d를 곱해가며 출력을 변경하는 ASR 모드와, ASR 모드의 방식을 그대로 따르면서 d값을 변경시켜 안정성을 강화한 Floating ASR 모드이다. ASR의 출력을 카운터로 사용하게 되면 CTR 모드보다 안정성이 높고 속도가 빠른 이점이 있다. 또한, CTR 모드에서 불편한 Random Access가 가능한 장점을 가지고 있으므로, Random Access가 필요한 부분에 넓게 사용될 수 있다.

ABSTRACT

This thesis suggests block cipher operating mode using ASR(Arithmetic Shift Register). ASR is called arithmetic shift register which is sequence that is not 0 but initial value A_0 multiplies not 0 or 1 but free number D on $GF(2^n)$. This thesis proposes ASR mode which changes output multiplying d and Floating ASR mode which has same function but having strengthened stability altering d. If we use ASR's output as a counter,there's advantage that it has higher stability and better speed than CTR. Also, ASR mode and FASR mode have advantage of Random access which is not being functioned on CTR mode,they can be widely used to any part which Random access is needed.

키워드

Block Cipher Mode, ASR, FASR, CTR

Ⅰ. 서 론

1990년대에 들어와 컴퓨터의 발달로 그동안 널리 사용되던 DES의 해독 가능성이 높아져 새로운 암호 알고리즘이 필요하게 되었고, 1997년 미국 상무 기술 표준국(National Institute of Standard Technology)에서 AES(Advance Encryption Standard)를 공모하였고, J. Deamen과 V. Rijemen이 제안한 Rijindael이 AES암호 알고리즘으로 채택되어 현재까지 쓰이고 있다[1].

AES는 지금까지 알려진 블록암호 알고리즘에 대한 모든 공격방법들에 대해 안전하도록 설계되었다[2]. 공격자들은 암호 알고리즘 자체를 공격하기보다는 그 알고리즘을 운영하는 운영모드에 대해서 공격을 하여 자신이 원하는 부분만을 해독하는 경우가 있다. 이와 같이 암호 알고리즘이 아닌 그 알고리즘을 운영하는 모드를 공격하는 경우가 있음에도 불구하고 많은 연구자들은 암호 알고리즘 자체만을 연구할 뿐 운영모드에 대한 연구는 활발히 진행되지 못하고 있는 실정이다.

현재까지 알려진 암호 알고리즘 운영모드는 ECB 모드(Electronic Code Book Mode), CBC 모드(Cipher Block Chaining Mode), CFB 모드(Cipher Feed Back Mode), OFB 모드(Output Feed Back Mode), CTR 모드(Counter Mode)가 있다[2][3][4][5]. 그러나 각각의 모드는 보안성, 암호문의 오류, 비트삽입 및 손실 등의 약점을 가지고 있다. 이에 각각의 모드를 개선하고 보안성에 더욱 강력한 모드의 개발이 필요하다.

본 논문에서는 ASR(Arithmetic Shift Register)[6]를 이용한 새로운 운용모드를 제안한다. 본 논문에서 제안하는 모드를 ASR 모드라 가칭하고, ASR 모드에서 d값을 변경시켜 안정성을 더욱 강화한 것을 Floating ASR 모드라고 가칭한다. ASR 모드의 운영방법은 ASR의 출력을 카운터로 사용하여 평문과 XOR 연산을 하는 방식으로 구현한다. 이는 LFSR(Linear Feedback Shift Register)을 카운터로 사용하여 카운터만을 암호화하는 CTR 모드보다 안정성이 높을 것으로 기대되며, CTR 모드 및 OFB 모드에서 불편한 Random Access가 더욱 편리해 질 것으로 예상된다.

본 논문의 구성은 2장에서 블록 암호 알고리즘의 운용 모드를 소개하고, 3장에서는 ASR을 소개한다. 4장에서는 ASR 모드와 Floating ASR 모드의 알고리즘을 C언어로 구현하고 암호화 및 복호화 과정을 살펴본다. 5장에서는 ASR 모드와 기존 모드의 암호화 및 복호화 과정의 속도를 테스트하여 결과를 비교 분석하며, 6장에서는 결론을 내린다.

II. 블록 암호의 운영모드

블록 암호는 고정된 크기의 블록만을 암호화한다. 따라서 평문의 길이가 블록 크기보다 클 경우 어떻게 블록 암호를 적용할 것인가를 해결하기 위해 다양한 응용 환경 아래에서 적절한 암호화 도구로 사용할 수 있는 여러 유형의 효율적인 운영 방식들을 제시하고 있다. 이것을 블록 암호화 모드(Block Cipher Mode)라고 한다. 블록 암호의 대표적인 운영모드에는 ECB 모드(Electronic Code Book Mode), CBC 모드(Cipher Block Chaining Mode), CFB 모드(Cipher Feed Back Mode), OFB 모드(Output Feed Back Mode), 그리고 CTR 모드(CounTeR Mode)가 있다.

첫 번째로 ECB 모드는 암호 운영방식 중 가장 간단한 방법으로 평문을 64비트씩 나누어 암호화 하는 방식이다. 이때 마지막 블록이 64비트가 되지 않을 때는 임의의 약속된 비트 모양을 패딩(Padding)하게 된다.

두 번째로 CBC 모드는 출력 암호문이 다음 평문 블록에 영향을 미치게 하여 각 암호문 블록이 전단의 암호문의 영향을 받도록 만든 방식으로 ECB모드에서 발생하는 동일한 평문에 의한 동일한 암호문이 발생하지 않도록 구성한 동작 모드이다.

세 번째로 CFB 모드는 CBC 모드와 마찬가지로 평문 블록이 동일한 경우 동일한 암호문이 나타나지 않도록 전단의 암호문이 다음 단의 평문에 영향을 미치게 구성하는 방식이다.

네 번째로 OFB 모드의 동작은 평문을 서로 독립적으로 암호화하는 ECB 모드의 단점과 오류 전파가 발생하는 CBC 모드와 CFB 모드를 개선한 동작 모드이다.

마지막으로 CTR 모드는 1씩 증가해가는 카운터를 암호화해서 키 스트림을 만들어내는 스트림 암호이다. 즉 카운터를 암호화한 비트열과 평문 블록과의 XOR 연산을 취한 결과가 암호문 블록이 된다. CTR 모드의 장점은 동일한 평문이 순서에 따라서 다른 암호문을 생성하기 때문에 안전도를 높일 수 있다는 것이다. 그러나 Counter의 특성상 대다수의 비트가 변화하지 않는 단점이 있다. CTR 모드의 수식 표현은 아래와 같다.

$$C_i = E(K, P_i \oplus (N \| T_i)) \quad (1)$$

III. ASR(Arithmetic Shift Register)

ASR(산술 시프트 레지스터)[6]이란 $GF(2^n)$상에서 0이 아닌 초기 값 A_0에 0 또는 1이 아닌 임의의 수 D를 곱하는 수열을 산술 시프트 레지스터(ASR-D, Arithmetic Shift Register-D)로 정의한다. 따라서 ASR-D의 i번째 값(상태) A_i는 A_0D^i가 된다. $GF(2^n)$상의 비복원 다항식 $P(x)$가 0 또는 1이 아닌 임의의 수 D에 대하여 '$D^k=1$'이 되는 t가 '$t=2^n-1$'로 유일하면 $P(x)$는 ASR-D의 특성다항식(characteristic polynomial)이 된다. 따라서 $GF(2^n)$에서 특성다항식으로 표현되는 ASR-D의 주기는 '2^n-1'이다. ASR-D를 C언어를 사용하여 구현하면 아래의 [표 1]과 같다.

표 1 $GF(2^{32})$상의 ASR-D의 C 프로그램

```
unsigned int P;//Characteristic polynomial
unsigned int A, D;
for (B=0 ; D ; D >>= 1)
  { if (D&1) B^=A;
    A=(A<<1)^((int)A>>31)&P);}
```

$GF(2^{32})$상의 ASR-2를 32비트 컴퓨터에서 구현하면 연산의 수가 종전의 갈로이 선형궤환 시프트 레지스터보다 33%, 피보나치 선형궤환 시프트 레지스터보다 71%적게 들어 효율적이다.

ASR-2의 선형복잡도는 n으로 종래의 궤환 시프트레지스터와 동일하며, ASR-D의 선형복잡도

는 '$n \le LC \le (n^2+n)/2$'으로 종래의 궤환 시프트 레지스터보다 높아서 안전도가 높다. ASR-2를 하드웨어로 구현을 하면 아래의 [그림 1]과 같다.

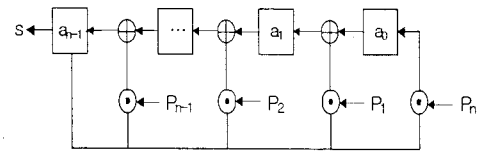

그림 1. ASR-2의 하드웨어 구현

IV. ASR을 이용한 암호 운용모드

4.1 ASR 모드

ASR 모드는 d를 곱해가면서 나타나는 ASR의 출력으로 키 스트림을 만드는 스트림 암호의 형태라 할 수 있다. ASR 모드는 ASR의 출력을 카운터로 사용하는 방식이다. 동작방법은 ASR의 첫 번째 출력 A_0가 초기벡터가 되며, A_0와 평문블록 M_1을 XOR 연산한 후 이를 암호기에 입력하여 초기화 키 K와 함께 암호화한다. 이때 ASR의 첫 번째 출력이 정해지면 두 번째, 세 번째 등은 첫 번째 출력으로부터 d를 곱하여서 출력을 구한다.

예를 들어 128 비트 블럭에서 ASR 특성방정식은 "00000002, 00000004, 00000005, 00000041"이고, d는 2^{19}을 선택할 수 있다. 이 모드는 블럭 마다 19 비트씩 회전하므로 변화하는 비트t 수가 많아서 CTR 모드보다 안정성이 높을 것으로 기대된다. 또한 A_0가 i번째 블록에 대한 A_i는 '$A_i = A_0 * D^{i}$'가 되므로 Random Access가 가능하다.

즉, ASR 모드는 식-2와 같이 정의되며, 그림으로 표현하면 그림- 2와 같이 나타낼 수 있다

$$
\begin{aligned}
&A_0 = IV \\
&A_i = A_{i-1} \times d \quad (i = 1,2,3,\cdots) \\
&C_j = E_k(M_j \oplus A_j) \quad (j = 0,1,2,3,\cdots) \\
&\text{(단, } A_0\text{는 초기 벡터 IV)}
\end{aligned}
\tag{2}
$$

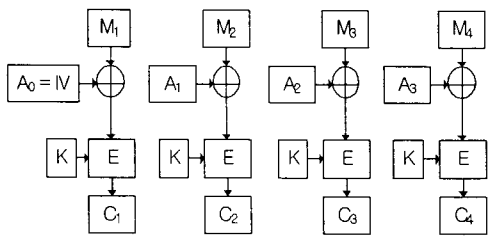

그림 2. ASR 모드

4.2 FASR(Floating ASR) 모드

FASR 모드는 평문의 값에 따라서, ASR의 d값을 변경시켜 안정성을 더욱 강화한 방식이다. 예로, 4.1절의 ASR 모드에서 제시한 특성 방정식에서 d는 "2^{19}", "2^{19}+1", "2^{13} ", "2^{13}+1"의 4가지 경우를 선택할 수 있다. 이와 같이 d를 선택하여 사용을 하게 되면, 안정성은 강화되지만 ASR의 장점인 Random Access가 불편해지는 단점이 있다.

FASR 모드는 식-3과와 같이 정의되며, 그림으로 나타내면 그림-2와 동일하다.

$$
\begin{aligned}
&A_0 = IV \\
&A_i = A_{i-1} \times d(M_{i-1} \cap 3) \quad (i = 1,2,3,\cdots) \\
&C_j = E_k(M_j \oplus A_j) \quad (j = 0,1,2,3,\cdots) \\
&\text{(단, } A_0\text{는 FASR의 첫 번째 출력)}
\end{aligned}
\tag{3}
$$

V. 구현 및 비교

IV장에서 제시한 ASR 모드와 FASR 모드를 각각 C언어로 구현한 내용을 아래의 표-2와 표-3에 보인다.

표 2. C로 구현한 ASR 모드

```
C = (word32 *)(kbuf + BUFP2) ;
wa = C[3]>>19 ;
C[3] = ((C[3]<<13) | (C[2]>>19) )^(wa<<1) ;
C[2] = ((C[2]<<13) | (C[1]>>19) )^(wa<<2) ;
C[1] = ((C[1]<<13) | (C[0]>>19) )^(wa<<2)^wa ;
C[0] = (C[1]<<13) ^ (wa<<6) ^ wa ;
```

표 3. C로 구현한 FASR 모드

```
C = (word32 *)(kbuf + BUFP2); a = C[3]>>i ;
if (pch & 0x8) i = 19 ;
    else i = 13 ;
if (pch & 0x80) b = -1 ;
    else b = 0 ;
C[3]=((C[3]<<(32-i)) | (C[2]>>i))^(a<<1)^ (C[3] &
b);
C[2] = ((C[2]<<(32-i)) | (C[1]>>i))^(a<<2)^ (C[2]
& b);
C[1] = ((C[1]<<(32-i)) | (C[0]>>i))^(a<<2)^ w ^
(C[1] & b);
C[0] = (C[1]<<(32-i))^(a<<6)^a^(C[0] & b);
```

CTR 모드는 Counter의 특성상 대다수의 비트가 변화하지 않는 단점이 있으므로, 변화하지 않는 비트를 공격할 가능성이 있다. 본 논문에서 제시한 ASR 모드와 FASR 모드는 각 블록마다 다수의 비트만큼 회전하므로 변화하는 비트수가 많아지므로 CTR 모드보다 안정성이 높을 것으로

예상된다.

다음으로 기존 암호운영모드와 ASR 모드의 수행속도를 비교하기 위해, 각 운영모드를 이용해서 암호화 해본다. 용량이 적은 파일은 수행속도 비교에 있어서 어려움이 있기 때문에, 용량이 큰 동영상 파일을 이용해서 각 모드의 수행 속도를 측정해 보도록 한다. 아래의 Test는 약 700MB정도의 크기를 가진 동영상 파일을 각 모드별로 암호화 하는데 걸리는 시간을 측정한 결과이다.

아래의 표-4는 WindosXP Professional OS가 설치되고, AMD Athlon 2600+ (1.91 GHz)의 CPU와 1GB의 Memory가 설치된 PC에서 Test한 결과이다. 표-4에서 ASR 모드의 수행시간이 기존의 모드와 비교했을 때 뒤지지 않고, 스트림암호 형식의 CTR 모드와 비교했을 때는 암호화 시간이 단축됨을 알 수 있다.

표 4. 각 모드의 수행시간 비교

(단위 : 초)

모드	암호화 시간
CBC	250.56300
CFB	248.90600
OFB	260.31200
CTR	267.45400
ASR	249.96900

또한, ASR 모드는 그림-2에서 알 수 있듯이 앞단 블록의 암호문이 다음 블록의 암호화과정에 영향을 미치지 않기 때문에 부분적으로 암/복호화가 가능하다. 이것은 큰 파일을 암/복호화 해야할 때, 유용하게 사용될 수 있다. 예를 들어서 1GB의 암호화된 파일을 읽으려 할 때에 필요한 부분을 알고 있다면, 그 부분만 복호화를 해서 필요한 정보를 얻는 것이 가능하다. 따라서, Random Access가 필요한 부분에서 ASR 모드를 사용할 수 있을 것으로 사료된다.

Ⅵ. 결 론

정보화 사회로 접어든 현재에 정보보호는 중요한 연구 분야로 자리 잡았다. 여러 가지 암호 알고리즘이 발명되고 또 이것을 공격하는 공격자들도 나타나고 있다. 따라서 공격자들의 공격으로부터 안전한 암호 알고리즘을 만드는 것이 암호학 연구의 가장 중요한 부분이다. 현재까지 알려져 있는 블록암호 알고리즘 중 AES는 공격자들의 공격으로부터 안전하게 설계되었다고 한다. 그러나 실제로 공격자들은 블록암호 알고리즘이 아닌 그 알고리즘을 운영하는 모드를 공격하여 자신이 원하는 정보를 경우가 있다. 그러나 블록암호 알고리즘의 연구보다 그 알고리즘을 운영하는 모드에 관한 연구는 미흡한 실정이다. 현재까지 알려진 모드들은 각각의 특징에 따라 보안상 취약한 부분들을 가지고 있다.

본 논문에서는 현재까지 알려진 블록암호 알고리즘의 운영모드보다 더욱 안정성이 강화된 모드들을 제안하였다.

ASR(Arithmetic Shift Register)을 이용한 블록암호 운영모드인 ASR 모드와 ASR 모드에서 d값을 변경시켜 안정성을 더욱 강화한 FASR모드가 그것이다.

ASR 모드의 암호화 과정은 평문블록과 초기화키를 ASR의 출력과 XOR하는 방식으로 구성되며, 식-2와 같이 '$C_j = E_k(M_j \oplus A_j)$'로 표현 가능하다. 이 모드는 대다수의 비트가 변하지 않는 CTR 모드의 단점을 보완하여, 각 블록마다 다수의 비트가 변화하는 구조를 가지므로 안정성이 더욱 향상되었을 뿐 아니라 작업 수행시간도 단축되었다. FASR 모드는 ASR모드의 암호화 과정을 그대로 따르면서, ASR의 d값을 변경시켜 안전성을 더욱 강화시켰다. 그러나 FASR 모드는 Random Access가 불편하게 구성이 되어있다.

제안한 ASR 모드는 안정성이 뛰어나고, Random Access가 가능할 뿐 아니라 수행속도 또한 빠르므로 앞으로 암호 알고리즘을 구성함에 있어 많이 사용될 수 있다.

참고문헌

[1] 서명룡, "하드웨어에 적합한 AES알고리즘에 관한 연구", 부경대학교 대학원 컴퓨터공학과 석사학위논문, 2006.

[2] 원동호, "현대 암호학", 도서출판그린, 2003.

[3] ETRI 부설 국가보안기술연구소, "현대암호학", 경문사, 2002.

[4] 김길호, "데이터 의존 셀룰라 오토마타를 사용한 블록 암호 알고리즘", 부경대학교 산업대학원 컴퓨터공학과 석사학위논문, 2002.

[5] Niels Ferguson, Bruce Schneier, "Practical Cryptography", John Wiley & Sons, Inc., 2003.

[6] 박창수, 조경연, "갈로이 선형 궤환 레지스터의 일반화", 전자공학회 논문지 제43권 CI편 제 1호, 2006