스트림 암호 ASC

김길호*, 송홍복**, 김종남* ,조경연*
*부경대학교 컴퓨터공학과
**동의대학교 전자공학과
e-mail : vnlqpcdd@hanmail.net

Stream Cipher ASC

Gil-Ho Kim*, Hong-Bok Song**, Jong-Nam Kim*, Gyeong-Yeon Cho*
*Dept of Computer Engineering, Pu-Kyong National University
**Dept of Electronics Engineering, Dong-Eui University

요 약

본 논문에서는 ASR(Arithmetic Shift Register)과 SHA-2로 구성된 32비트 출력의 새로운 스트림 암호 ASC를 제안한다. ASC는 소프트웨어 및 하드웨어 구현이 쉽게 디자인된 스트림 암호 알고리즘이다. 특히 계산능력이 제한된 무선 통신장비에서 빠르게 수행할 수 있도록 개발되었다. ASC는 다양한 길이(8-32바이트)의 키를 지원하고 있으며, 워드 단위로 연산을 수행한다. ASC는 매우 간결한 구조를 가지고 있으며 선형 궤환 순서기(Linear Feedback Sequencer)로 ASR을 적용하였고, 비선형 순서기(Nonlinear sequencer)로 SHA-2를 적용하여 크게 두 부분으로 구성되어 있는 결합 함수(combining function) 스트림 암호이다. 그리고 8비트, 16비트, 32비트 프로세스에서 쉽게 구현이 가능하다. 제안한 스트림 암호 ASC는 최근에 표준 블록 암호로 제정된 AES, ARIA, SEED등의 블록 암호보다는 6-13배 빠른 결과를 보여주고 있으며, 안전성 또한 현대 암호 알고리즘이 필요로 하는 안전성을 만족하고 있다.

1. 서론

대칭 키(Symmetric Key) 암호 알고리즘의 한 부분인 스트림 암호(Stream Cipher)는 1970년대 유럽을 중심으로 하드웨어 구현이 용이한 LFSR 기반의 이진 수열 발생기를 이용하여 평문과 이진 수열 발생기에서 생성된 이진 수열과 XOR연산을 수행하여 이진 수열로 된 암호문을 만드는 암호 알고리즘이다. 1990대 이후 암호화 기술이 일반화되고 인터넷을 통한 대용량의 멀티미디어 데이터 전송의 증가로 인한 빠른 암호 알고리즘의 개발이 필요 했다. 일반적으로 스트림 암호는 블록 암호(Block Cipher)보다 5-10배 정도 빠르게 실행되는 장점을 가진다. 그리고 스트림 암호는 비트단위로 암호화 하므로 에러 전파(error propagation) 현상이 없다. 최근에는 스트림 암호 또한 블록 암호와 마찬가지로 블록 단위로 키를 생성하여 암호화하는 방식이 널리 사용되고 있다. 1990년대 후반 소프트웨어 구현이 용이한 스트림 암호가 등장하기 시작 했고, 특히 2000년대에 유럽의 NESSIE(New European Schemes for Signatures, Integrity, and Encryption)[1], 일본의 CRYPTREC(Cryptography Research and Evaluation Committees)[2] 등의 국제적인 암호 공모사업으로 스트림 암호도 공모되어 여러 종류의 새로운 스트림 암호가 제안 되었다.

휴대폰과 같은 개인용 무선 통신장비에서 데이터에 대한 암호의 적용은 몇 가지 고려해야할 사항이 있다. 첫째 계산능력의 한계로 인해 공개 키 암호와 같은 많은 계산능력이 요구되는 알고리즘은 적당하지 않다. 둘째 무선통신은 높은 오차율(error rates)로 인해 모드(mode)가 적용되는 블록 암호는 오차의 전파(propagation) 때문에 무선통신용 암호 알고리즘으로 적당하지 않다. 셋째 부족한 대역폭(shortage bandwidth) 때문에 휴대폰에서 베이스 스테이션(base station)으로 암호의 적용은 많은 끊김(delay) 현상을 발생 시킨다. 그래서 블록 암호보다 더욱 빠른 고속 암호 알고리즘이 필요하다. 마지막으로 메모리를 적게 소비하는 소프트웨어지향(software oriented) 암호 알고리즘이 필요하다. 하드웨어로 암호 알고리즘의 구현은 휴대폰 단말기의 추가적인 가격(cost) 상승의 요인이 된다. 이와 같은 이유로 인해 휴대폰과 같은 개인용 무선 통신장비에서의 암호 알고리즘의 적용은 빠른 수행이 가능하고 안전성이 검증된 소프트웨어 구현의 스트림 암호 알고리즘이 적당하다.

본 논문에서는 ASR(Arithmetic Shift Register)[3]과 SHA-2[4]로 구성된 32비트 출력의 새로운 스트림 암호 ASC를 제안한다. ASC는 소프트웨어 및 하드웨어 구현이 쉽게 디자인된 스트림 암호 알고리즘이다. 특히 계산능력이 제한된 무선 통신장비에서 빠르게 수행할 수 있도록 개발되었다. ASC는 다양한 길이(8-32바이트)의 키를 지원하고 있으며, 워드 단위로 연산을 수행한다. ASC는 매우 간결한 구조를 가지고 있으며 선형 궤환 순서기(Linear Feedback Sequencer)로 ASR을 적용하였고, 비선형 순서기(Nonlinear sequencer)로 SHA-2를 적용하여 크게 두 부분으로 구성되어 있는 결합 함수(combining

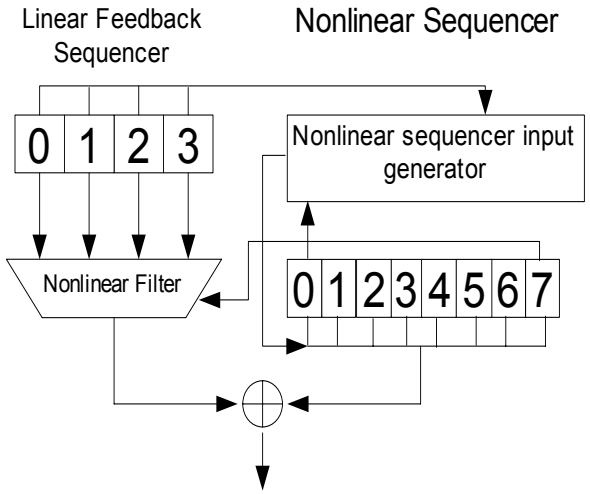
function)[5] 스트림 암호이다. 그리고 8비트, 16비트, 32비트 프로세스에서 쉽게 구현이 가능하다.

제안한 스트림 암호 ASC는 최근에 표준 블록 암호로 제정된 AES, ARIA, SEED등의 블록 암호보다는 6-13배 빠른 결과를 보여주고 있으며, 안전성 또한 현대 암호 알고리즘이 필요로 하는 안전성을 만족하고 있다.

본 논문의 구성은 2장에서 ASC 구조를 설명하고, 3장에서 ASC를 소프트웨어 구현과 수행 결과를 설명하며, 4장에서 ASC의 안전성에 대한 설명하고, 마지막으로 결론으로 끝맺는다.

2. ASC 구조

스트림 암호 ASC는 ASR과 SHA-2로 구성한다. ASR과 SHA-2는 워드 단위로 구성되어 선형 2진 연산을 처리하고 최종적으로 32비트 키 스트림을 생성한다. (그림 1)은 ASC의 전체적인 흐름을 그림으로 표현한 것이다.



(그림 1) ASC의 전체 구성도

2.1 초기화

ASC는 8-32바이트의 가변적인 키를 가지며 SHA-2 해시 알고리즘을 사용하여 초기화를 수행한다. 초기화 과정은 다음과 같다.

1. SHA-2의 메시지 버퍼 32 바이트(512 비트)를 '0'으로 초기화한다.

2. 입력된 N비트 키를 메시지 버퍼 (N-1)-0 비트에 복사한다.

3. 메시지 버퍼의 511-480 비트에 32비트 N값을 저장한다.

4. SHA-2 알고리즘을 적용하여 256 비트 해시(hash)를 만든다. 생성된 256 비트의 해시값은 (그림 1)의 첫 번째 0-7워드에 복사하여 비선형 순서기(SHA-2)의 초기 값으로 한다.

5. '4'항에서 생성된 256 비트 해시를 메시지 버퍼 256-511 비트에 복사한다.

6. SHA-2 알고리즘을 적용하여 256비트 해시(hash)를 만든다. 생성된 256 비트의 해시값의 127-0 비트를 ASR의 초기값으로 한다.

2.2 ASR

의사난수발생기로 사용할 수 있는 산술 쉬프트 레지스터(Arithmetic Shift Register)는 $GF(2^n)$상에서 0이 아닌 초기값에 0 또는 1이 아닌 임의의 수 D를 곱하는 수열로 정의한다. ASR의 i번째 값(상태) A_i는 $A_0 \cdot D^i$가 된다.

D^k = 1이 되는 t가 $t = 2^n-1$로 유일하게 되는 비복원 다항식(irreducible polynomial)이 ASR의 특성다항식(Characteristic Polynomial)이며, ASR의 주기는 2^n-1로 최대 주기를 가진다. 그리고 ASR의 선형 복잡도(Linear Complexity)는 기존의 LFSR(Linear Feedback Shift Register)의 선형 복잡도 보다 높아서 안전도가 높다.

본 논문에서는 $GF(2^{128})$상에서 특성다항식은 '0x1 00000001 00000002 00000004 00000013', $D = 2^{19}$을 적용한다. ASR의 동작을 소프트웨어로 작성하면 다음과 같다.

```
w0=ASR[3]》13;
ASR[3]=((ASR[3]《19)|(ASR[2]》13))⊕w0;
ASR[2]=((ASR[2]《19)|(ASR[1]》13))⊕(w0《1);
ASR[1]=((ASR[1]《19)|(ASR[0]》13))⊕(w0《2);
ASR[0]=(ASR[0]《19)⊕(w0《4)⊕(w0《1)⊕w0;
```

2.3 비선형 필터

비선형 필터는 메모리가 필요 없는 함수로 워드 단위로 처리하는 ASR에서 1워드(32비트)를 생성하는 함수이다. 알고리즘은 다음과 같다.

```
w0=ASR[3]+ASR[0];
w0=ROTL(w0,16);
if(SHA[7]&4)
  w0+=ASR[2];
else
  w0+=ASR[2]+ASR[0];
w0+=(ASR[1]⊕ASR[2])+(SHA[7]&1);
i=SHA[7]》29;
w0=SHA[i]+ROTL(SHA[i⊕4],16)⊕w0;
```

ROTL()은 왼쪽 회전연산(rotate)을 수행하는 매크로 함수이고 배열 ASR의 전체 4워드(128비트)에서 1워드의 선택과정은 배열 SHA의 마지막 워드의 MSB(most significant bit) 3비트를 이용하여 최종적으로 32비트 출력을 생성한다.

2.4 비선형 순서기 입력 발생기

비선형 순서기 입력 발생기는 SHA의 첫 번째 워드에서 LSB(least significant bit) 2비트만을 이용하여 ASR의

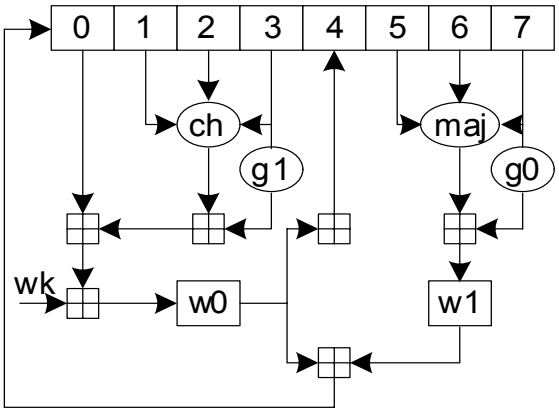
4개의 워드 중 하나를 선택한다. 그리고 선택된 ASR 워드를 8비트 회전연산을 수행 후 고정된 상수 P와 XOR연산을 수행한다. 다음은 비선형 순서기 입력 발생기의 알고리즘이다.

```
i=SHA[0]&3;
wk=ROTL(SHA[i],8)⊕P;
```

wk는 32비트 출력 변수이고, P는 고정된 상수 0xb7e15163이다. 32비트값 wk는 비선형 순서기에서 사용된다.

2.5 비선형 순서기

비선형 순서기는 SHA-2의 해시 함수를 조금 변형시킨 것으로 비선형 변환을 한다. SHA-2는 안전성이 검증된 표준 해시 함수이다. (그림 2)은 비선형 순서기의 전체적인 흐름도를 나타낸다.



(그림 2) SHA-2 흐름도

(그림 2)와 같은 과정을 수행하여 256비트(8워드)의 새로운 SHA-2값을 얻을 수 있다. (그림 3)에서 w0, w1은 32비트 변수이고, 함수 ch, maj는 3워드를 입력 받아 1워드를 출력하는 비선형 함수이고, 함수 g0, g1은 1워드를 입력 받아 고정된 길이의 회전 연산을 수행하는 선형 변환 함수이다. 함수 ch, maj, g0, g1은 다음과 같다.

```
#define ch(x,y,z) (((x)&(y))⊕(~(x)&(z)))
#define maj(x,y,z) (((x)&(y))⊕((x)&(z))⊕     \
                    ((y)&(z)))
#define g0(x) (ROTR(x,2)⊕ROTR(x,13)⊕         \
                              ROTR(x,22))
#define g1(x) (ROTR(x,6)⊕ROTR(x,11)⊕         \
                              ROTR(x,25))
```

ROTR()은 오른쪽 회전연산을 수행하는 매크로 함수이다. ~는 비트별 NOT연산자 이다. (그림 3)에서 wk는 2.3절의 비선형 순서기 입력 발생기에서 생성된 32비트 값으로 SHA-2에서는 32비트 키 값으로 사용된다.

3. ASC의 구현 및 수행결과

제안한 ASC의 소프트웨어 구현은 Visual Studio 2005 C 컴파일러를 사용하여 암호와 복호가 정상적으로 수행되는 것을 확인했으며, 약 30MB 정도의 그림, 표, 특수문자 등이 있는 일반적인 한글 문서파일로 Windows Vista, Intel Core(TM)2 Duo CPU 2.26Ghz, 2.27Ghz, 2GB RAM의 환경에서 기존의 여러 암호 알고리즘과 제안한 알고리즘의 수행 시간을 테스트했다. 결과는 <표 1>과 같다.

<표 1> 수행시간 테스트 결과

시간 / 알고리즘	암호, 복호 수행 결과
ASC	100
SSC2	79
AES	842
ARIA	1283
SEED	604

제안한 알고리즘 ASC의 수행 시간을 100으로 환산했을 때 Carroll, Chan 그리고 Zhang[6]가 제안한 32비트 스트림 암호 SSC2 약 79로 ASC보다는 빠르다. 그러나 SSC2는 Hawkes, Rose 그리고 Quick[7]의 논문에 의하면 LFG(Lagged Fibonacci Generator)[8]의 짧은 주기와 상관관계 분석(Correlation Analysis)[9]을 통해 현대 암호에서 필요로 하는 안전성을 만족 시키지 못한다고 주장 했다. 나머지 AES[10], ARIA[11], SEED[12]는 128비트 블록 암호이다. 제안한 ASC 스트림 암호와 최소 6-13배 정도 차이가 난다.

4. ASC의 안전성 검증

제안한 ASC 스트림 암호 알고리즘은 크게 선형 궤환 순서기(ASR)와 비선형 순서기(SHA-2)로 나눌 수 있다. ASR은 선형 함수로 최대 주기를 갖고, 선형 복잡도를 계산할 수 있다[3]. 그러나 SHA-2는 비선형 함수로 주기는 계산할 수 없지만 특정한 키에 대해서 평문에 대응되는 암호문쌍을 전수조사를 할 경우 최소 2^{256}이 있으면 특정한 키를 찾을 수 있다. 이를 의사 선형 복잡도(Pseudo Linear Complexity)라 할 때 ASR과 SHA-2는 서로가 독립적으로 결합한 부울함수(boolean function) f : $GF(2^n)$ -> $GF(2^{32})$에 의하여 결합하여 출력하는 논리이다. 여기서 함수 f는 결합 함수라 하고 XOR연산을 적용한다.

제안한 알고리즘의 안전성 분석은 결합 함수를 거친 n비트 결합 수열(combined sequence)의 선형 복잡도는 n비

트 수열의 합과 곱에 대한 선형 복잡도를 반복적으로 적용해서 계산할 수 있다. ASR, SHA-2의 출력을 각각 α, β라 하고 α, β가 이진 수열일 때 각 수열 항을 GF(2)상에서 더한 이진 수열은 α + β가 되고, 각 수열의 항을 GF(2)상에서 곱한 이진 수열은 αβ가 된다.

특히 β는 의사 선형 복잡도가 2^{256}일 때 만일 공격자가 ASR의 출력 α를 완벽하게 알고 있다고 할지라도 β의 출력을 알기 위해서는 최소 2^{256}개의 평문과 이에 대응되는 암호문쌍을 조사해야 한다. 그래서 최종적으로 ASC의 의사 선형 복잡도는 최소 $(2^{128}-1)*2^{256}$이 된다.

5. 결론

본 논문에서는 크게 ASR과 SHA-2로 구성된 32비트 출력의 새로운 스트림 암호 ASC를 제안한다. ASC는 소프트웨어 구현이 쉽게 디자인된 스트림 암호 알고리즘이다. 특히 계산능력이 제한된 무선 통신장비에서 빠르게 수행할 수 있도록 개발되었다. ASC는 다양한 길이(8-32바이트)의 키를 지원하고 있으며, 워드 단위로 연산을 수행한다. ASC는 매우 간결한 구조를 가지고 있으며 선형 궤환 순서기로 ASR을 적용하였고, 비선형 순서기로 SHA-2를 적용하여 크게 두 부분으로 구성되어 있는 결합 함수 스트림 암호이다. 그리고 8비트, 16비트, 32비트 프로세스에서 쉽게 구현이 가능하다.

제안한 스트림 암호 ASC는 SSC2보다 수행 속도는 느리지만 최근에 표준 암호로 제정된 AES, ARIA, SEED 등의 블록 암호보다는 6-13배 빠른 결과를 보여주고 있으며, 안전성 또한 현대 암호 알고리즘이 필요로 하는 안전성을 만족하고 있다. 그래서 제안한 스트림 암호 ASC는 휴대폰과 같은 제한된 리소스를 가지고 있는 무선 통신장비에서 사용할 암호 알고리즘으로 적당하다.

감사의 글

본 연구는 교육과학기술부와 한국산업기술재단의 지역혁신 인력양성사업, 중소기업청의 산학연공동기술 개발 지원 사업(선도형)의 지원으로 수행되었음.

참고문헌

[1] “New European Schemes for Signatures. Integrity. and Encryption(NESSIE)” http://cryptonessie.org/.

[2] “Cryptography Research and Evaluation Committees(CRYPTREC)” http://www.cryptrec.go.jp/.

[3]박창수, 조경연 “갈로이 선형 궤환 레지스터의 일반화” 전자공학회논문지 제43권 C1편 제1호 2006.1.

[4] NIST, *Secure hash standard*, FIPS-180-2, 2002.

[5] L. Brynielsson "On the linear complexity of combined shift register sequences" In F. Pichler editor Advances in Cryptology - Eurocrypt '85 p.p 156-166 Berlin Springer-Verlag 1986.

[6] C. Carroll, A. Chan, and M. Zhang "The software-oriented stream cipher SSC-Ⅱ" FSE 2000 LNCS Vol.1978 p.p 39-56 2000.

[7] P. Hawkes, F. Quick, and G. Roes "A practical cryptanalysis of SSC2" Selected Areas in Cryptography 2001 LNCS Vol. 2259 p.p 27-37 2001.

[8] D. E. Knuth "The Art of Computer programming. Volume 2 : Seminumerical Algorithms" 3rd Edi tion Addison-Wesley 1997.

[9] P. Hawkes, and G. Rose "Correlation cryptanalysis of SSC2" Presented at the Rump Session of CRYPTO 2000.

[10] J. Daemen, and V. Rijmen, "AES Proposal : Rijndael" http://www.csrc.nist.gov/encryption/aes/rijndael/Rijndael.pdf, 1999.

[11] ARIA, http://www.nsri.re.kr/ARIA/.

[12] SEED, http://www.kisa.or.kr/seed/.