소프트웨어 구현에 적합한 고속 스트림 암호 AA32

정회원 김 길 호*, 박 창 수*, 김 종 남**, 조 경 연*

Fast Stream Cipher AA32 for Software Implementation

Gil-Ho Kim*, Chang-Soo Park*, Jong-Nam Kim**, Gyeong-Yeon Cho* ***Regular Members***

요 약

스트림 암호는 블록 암호보다 안전성은 떨어지지만 수행 속도가 빠른 것이 큰 장점이었다. 그러나 최근까지 블록 암호의 수행 속도를 개선한 알고리즘 개발로 지금은 AES의 경우 스트림 암호와 수행 속도 차가 거의 없게 되어, 안전하면서 빠른 스트림 암호 개발이 절실히 요구된다. 본 논문에서는 ASR(Arithmetic Shift Register)과 간단한 논리연산으로 구성된 32비트 출력의 고속 스트림 암호 AA32를 제안한다. 제안한 알고리즘은 소프트웨어 구현이 쉽게 디자인된 스트림 암호 알고리즘으로 128비트 키를 지원하고 있으며, 워드와 바이트 단위로 연산을 수행한다. AA32의 전체 구성은 선형 궤환 순서기(Linear Feedback Sequencer)로 ASR 151비트를 적용하였고, 축소함수는 비선형(Non-Linear) 연산을 위한 S-박스를 사용하지 않고 간단한 논리연산을 사용한 크게 두 부분으로 구성되어 있는 매우 간결한 구조의 스트림 암호이다. 제안한 스트림 암호 AA32는 SSC2, Salsa20 보다 수행 속도 테스트결과 빠른 결과를 보여주고 있으며, 안전성 또한 현대 암호 알고리즘이 필요로 하는 안전성을 만족하고 있다. 제안한 암호 알고리즘은 휴대폰과 같은 무선 인터넷 환경과 DRM(Digital Right Management) 등과 같은 실시간 처리가 필요한 분야와 제한된 환경인 무선 센서 네트워크(Wireless Sensor Network)에 사용 가능한 고속 스트림 암호 알고리즘이다.

Key Words : AES, LFSR, ASR, SSC2, Salsa20, DRM, WSN

ABSTRACT

Stream cipher was worse than block cipher in terms of security, but faster in execution speed as an advantage. However, since so far there have been many algorithm researches about the execution speed of block cipher, these days, there is almost no difference between them in the execution speed of AES. Therefore an secure and fast stream cipher development is urgently needed. In this paper, we propose a 32bit output fast stream cipher, AA32, which is composed of ASR(Arithmetic Shifter Register) and simple logical operation. Proposed algorithm is a cipher algorithm which has been designed to be implemented by software easily. AA32 supports 128bit key and executes operations by word and byte unit. As Linear Feedback Sequencer, ASR 151bit is applied to AA32 and the reduction function is a very simple structure stream cipher, which consists of two major parts, using simple logical operations, instead of S-Box for a non-linear operation. The proposed stream cipher AA32 shows the result that it is faster than SSC2 and Salsa20 and satisfied with the security required for these days. Proposed cipher algorithm is a fast stream cipher algorithm which can be used in the field which requires wireless internet environment such as mobile phone system and real-time processing such as DRM(Digital Right Management) and limited computational environments such as WSN(Wireless Sensor Network).

※ 본 연구는 교육과학기술부와 한국산업기술진흥원의 지역혁신인력양성사업, 중소기업청의 산학연공동기술지원사업으로 수행된 연구결과임.

* 부경대학교 IT융합응용공학과 마이크로프로세서 연구실(vnlqpcdd@hanmail.net),

** 부경대학교 IT융합응용공학과 영상 및 멀티미디어 통신연구실(jongnam@pknu.ac.kr).

논문번호 : KICS2010-04-148, 접수일자 : 2010년 4월 2일, 최종논문접수일자 : 2010년 6월 3일

Ⅰ. 서 론

대칭 키 암호(Symmetric Key)는 크게 블록 암호(Block Cipher)와 스트림 암호(Stream Cipher)로 나눌 수 있다. 스트림 암호는 선형 궤한 시프트 레지스터(Linear Feedback Shifter Register)를 이용한 최대 주기 수열과 비선형 여과기(Non-linear Filter Generator)를 결합하여 선형 복잡도(Linear Complexity)가 큰 수열을 얻는 알고리즘을 많이 사용한다.

LFSR로서는 난수성을 갖는 좋은 통계적 특성(Statistical Properties)과 최대 주기 수열을 생성하는 의사 랜덤 수열 생성기(Pseudo-Random Number Generator)로는 이산대수(Discrete Logarithm) PRNG[1], 2차 잉여(Quadratic residuosity) PRNG[2], 일방향 함수(One Way Functions) PRNG[3], 혼돈(Chaos) PRNG[4], 피보나치(Fibonacci) PRNG[5] 등이 있고, 이와 같은 PRNG를 여러 개 결합하여 스트림 암호를 구성 할 수 있으며 다른 방식으로는 PRNG와 비선형 변환을 하는 NFG를 결합하는 방식이 있다. NFG는 미리 계산 후 메모리에 적재한 다음 참조하는(Look-Up Table: S-box)[6] 방식과 계산 복잡도가 높은 곱셈, 나눗셈, 지수연산 등의 비선형 산술 연산 및 유한체 연산을 일반적으로 많이 사용하고 있다.

현대 암호는 정치, 군사적인 목적의 비밀통신 보다는 인터넷 기반의 일반적인 경제, 사회 활동의 안전성, 신뢰성, 개인 정보보호 등을 위한 핵심 기술로 인식되고 있으며, 특히 비밀통신 및 DRM과 같은 실시간 처리가 필요한 다양한 분야에 적용하기 위해서 암호 알고리즘의 빠른 수행 속도의 중요성이 더욱 증대 되고 있다. 스트림 암호는 블록 암호보다 안전성은 떨어지지만 수행 속도가 5~10배 정도 빠른 장점을 가지고 있다. 그러나 최근 들어 블록 암호 알고리즘의 수행 속도 개선을 위한 많은 연구 결과로 AES[7,8]의 경우 스트림 암호와 수행 속도 측면에서 거의 차이가 없게 되었다. 그래서 안전하면서 휴대폰과 같은 무선 인터넷 환경과 DRM등과 같은 실시간처리가 필요한 분야와 제한된 환경인 무선 센서 네트워크에 필요한 고속 스트림 암호의 개발이 절실히 요구 된다.

고속 스트림 암호의 설계를 위한 방법으로는 첫째 여러 PRNG를 기반으로 한 최대 주기 수열을 만드는 LFSR의 수행 속도를 향상 시키는 방법으로 비트 단위로 피드백 되는 일반적인 LFSR을 현재는 워드 단위 피드백을 통해 소프트웨어 구현에서 상당히 빠른 결과를 보여주고 있다. 둘째 비선형 변환을 위한 낮은 계산 복잡도를 가지는 논리 연산의 사용으로 계산 복잡도가 높은 유한체 연산, 곱셈, 나눗셈, 지수연산 등과 같은 산술 연산보다 빠르게 수행할 수 있다. 마지막으로 S-박스를 사용하지 않고 비선형 연산을 구현 하는 것이 스트림 암호의 수행 속도를 향상 시킬 수 있다. 특히 소프트웨어 및 하드웨어 환경이 제한적인 무선 센서 네트워크에서 큰 S-박스 적용은 불가능하다.

위와 같은 고속 스트림 암호 설계를 바탕으로 제안한 AA32는 매우 간결한 구조의 32비트 출력을 만드는 스트림 암호이며, 구성은 LFSR로 잘 알려지지는 않았지만 151비트 산술 쉬프트 레지스터(Arithmetic Shift Register)[9]를 적용하였고, 비선형 변환을 위한 함수는 간단한 논리 연산을 적용한 축소함수(Reduction Function)를 사용한 크게 두 부분으로 구성된 결합 함수(Combining Function)스트림 암호이다[10].

151비트의 산술 쉬프트 레지스터(ASR)는 기존의 LFSR보다 선형 복잡도가 높으며, 수행 속도 또한 워드 단위로 피드백 되므로 빠르다. 그리고 비선형 변환을 위한 축소함수는 S-박스를 사용하지 않고 간단한 논리 연산인 AND, OR, XOR, 회전연산(Rotate)만을 사용하여 안전성과 수행 속도를 빠르게 개선하였다.

제안한 AA32는 SSC2[11], Salsa20[12] 보다 수행 속도 테스트결과 최소 67%에서 최대 300% 정도 빠른 결과를 보여주고 있으며, 안전성 또한 현대 암호 알고리즘이 필요로 하는 안전성을 만족하고 있다.

본 논문의 구성은 2장에서 무선통신 및 무선 센서 네트워크를 위해 개발된 여러 스트림 암호와 전반적인 스트림 암호에 대해서 간략한 소개와 문제점을 중심으로 설명하고, 3장에서 제안한 AA32 알고리즘을 자세히 설명하고, 4장에서 AA32의 소프트웨어 구현 결과 및 안전성에 대해 분석하고, 마지막으로 결론으로 끝맺는다.

Ⅱ. 관련연구

스트림 암호는 블록 암호와 다르게 국제적인 공모를 통해 현재까지 표준을 정하지는 못했다. 대표적인 국제 공모사업으로는 NESSIE(New European Schemes for Signatures, Integrity, and Encryption)[13], ecrypt Ⅱ[14]가 있었으며 특히 NESSIE 프로젝트에서 제안된 스트림 암호는 최종 평가에서 수행속도가 만족되지 못해 최종적으로 선택된 알고리즘 없이 종료 하였고, 다음으로 진행된 ecrypt Ⅱ에서는 스트림 암호의 연구만을 위한 프로젝트로 진행 되었다. 현재까지 진행된 국제공모사업에서 스트림 암호가 선정되지 못한 주된 원인은 스트림 암호의 수행 속도가 블록 암호보다 월등히 앞서지 못 했다는 것이다. 이는 LFSR기반의 스트림 암호의 한

계를 보여 주는 것이며 그 대안으로 AES와 같은 빠른 블록 암호의 운영모드(OFB, CTR, CFB 등)를 이용한 방법이 있다.

이와 같은 상황에서 앞으로의 스트림 암호의 개발 방향은 Gbps급 초고속 스트림 암호나 초경량 하드웨어 환경에 맞는 스트림 암호의 개발이 절실히 요구되고 있다. 그러나 국제적인 표준으로 정해진 스트림 암호는 없지만 특별한 분야별 표준은 다음과 같다. GSM[15]: A5/1, A5/2, A5/3, IEEE 802.11의 WEP: RC4[16], UMTS/3GPP: f8[17], 블루투스: E0[18] 등이다.

전 세계적으로 가장 많이 사용되고 있는 개인 휴대 통신 기술인 GSM은 A5 알고리즘을 사용한다. A5/1[19], A5/2는 현재 분석되어 안전성에 위협을 주고 있으며, A5의 대안으로 개발된 고속 소프트웨어 구현을 위한 SSC2는 Hawkes, Rose 그리고 Quick[20]의 논문에 의하면 LFG(Lagged Fibonacci Generator)[21]의 짧은 주기와 상관관계 분석(Correlation Analysis)[22]을 통해 현대 암호에서 필요로 하는 안전성을 만족 시키지 못한다고 주장 했다.

그리고 무선 LAN 표준을 정의하는 IEEE 802.11 규약의 일부분으로 무선 LAN 운영간 보안을 위해 사용되는 WEP는 RC4를 사용한다. RC4는 1987년 개발된 스트림 암호로 바이트 단위 치환을 기본 동작으로 내부 상태를 갱신하는 방식으로 1994년 역어셈블리 방식으로 그 구조가 인터넷에 공개된 후 많은 암호학자들로부터 관심을 받았다. 현재까지 안전하다고 여겨지고 있지만 출력 키수열과 랜덤함수를 구별하는 distinguishing 공격[23]과 WEP에 적용되는 경우 재동기 과정[24]에서 문제점이 발견되었다. 그리고 RC4는 256바이트의 메모리를 사용하므로 초경량 하드웨어 환경에서는 적합하지 않다.

블루투스 무선 환경에서 사용되는 스트림 암호 E0는 합이 128비트인 4개의 LFSR로 이루어져 있으며 64비트 키를 사용한다. 현재까지는 안전하지만 많은 분석 결과들이 발표되어 안전성에 위협이 되고 있다.

Ⅲ. 제안한 AA32

현대 암호 알고리즘의 안전성을 만족하면서 수행 속도가 빠른 스트림 암호의 설계는 쉬운 일이 아니며, 특히 제한적인 소프트웨어 및 하드웨어 환경인 WSN에서 구현은 더욱 어렵다.

AA32는 LFSR 역할을 하는 ASR과 비선형 여과기(NFG)로 축소함수를 사용하여 크게 두 부분으로 구성된 고속 스트림 암호 알고리즘이다. 그림 1은 AA32의 전체적인 흐름 구성도를 표현한 것이다.

그림 1의 진행과정은 초기화 과정을 수행한 다음 151비트의 ASR을 업 데이터 한 후 ASR의 워드 또는 5비트 단위로 축소함수와 함께 적용하여 최종적으로 32비트의 키를 생성한다. 이어지는 다음 절부터 그림 1의 순서대로 진행과정을 자세히 설명한다.

그리고 본 논문에서 그림이나 알고리즘 등에서 사용할 변수 이름은 영어 대문자인 경우는 워드 단위이고, 소문자인 경우는 바이트 단위이다. 그리고 변수명에 위첨자는 상태값을 나타내며, 아래첨자는 워드 또는 바이트 단위 순서 번호이다. 예를 들어 AES^0_3과 b^0_3은 각각 AA32의 초기상태(0)에서 128비트와 32비트의 마지막 워드와 바이트를 각각 나타낸다. 이와 같은 표기법은 ASR도 같이 적용한다. 그리고 논리 및 산술 연산은 일반적인 연산자 표기방법을 그대로 적용한다.

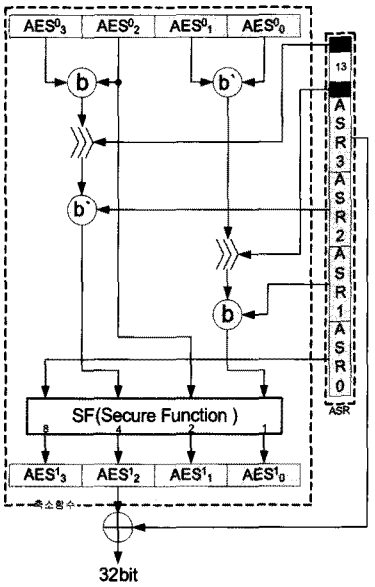


그림 1. AA32의 전체 구성도

3.1 초기화 과정

초기화 과정은 ASR 151비트(32비트 4개와 마지막 최상위 워드는 23비트만 사용)와 축소함수 128비트의 초기상태를 만드는 과정으로 스트림 암호에서는 최종적인 키 생성 과정에서 초기상태가 굉장히 중요하다[25]. 본 논문에서는 블록 암호인 AES를 이용하여 AA32의 279비트(151+128)를 초기화 시킨다. AES를 이용한 초기화 방법은 AES의 안전성에 대한 보장만큼 초기화의 안전성을 보장 받게 된다. 구체적인 초기화 방법은 다음과 같다.

먼저 사용자가 입력한 128비트 키와 128비트

IV(Initial Vector)를 각각 AES의 마스터 키와 평문으로 두고 AES의 10라운드를 적용한 후 그림 1의 축소함수 초기값인 AES^0_0 ~ AES^0_3에 대입한다. 그리고 AES 8, 9 라운드의 256비트 결과값을 순차적으로 ASR의 151비트에 대입한다. 이때 ASR^0_4은 23비트만 유효한 값이 된다. 그리고 ASR^0_0 ~ ASR^0_4을 ASR의 다음 상태값인 ASR^1_0 ~ ASR^1_4으로 그림 2와 같은 방법으로 업데이터 한 값이 최종적인 AA32의 ASR 151비트의 초기값이 된다.

만일 ASR^1_0 초기값이 0이거나, ASR^1_4의 23비트가 0 또는 1인 경우는 초기화 과정을 현재의 값을 가지고 AES 알고리즘을 10라운드 더 수행한 후 처음부터 다시 적용한다.

3.2 ASR

PRNG로 사용할 수 있는 산술 쉬프트 레지스터(ASR)는 $GF(2^n)$상에서 0이 아닌 초기값에 0 또는 1이 아닌 임의의 수 D를 곱하는 수열로 정의한다. ASR의 i번째 값(상태) ASR^i는 $ASR^0 \cdot D^i$가 된다. D^k = 1이 되는 k가 $t = 2^n - 1$로 유일하게 되는 비복원 다항식(Irreducible Polynomial)이 ASR의 특성다항식(Characteristic Polynomial)이며, ASR의 주기는 $2^n - 1$로 최대 주기를 가진다. 그리고 ASR의 선형 복잡도는 기존의 LFSR의 선형 복잡도 보다 높아서 안전도가 높다.

본 논문에서는 $GF(2^{151})$상에서 특성다항식은 16진수로 0x00800000 0x00000001 0x00000001 0x00000004 0x00000025, $D = 2^{23}$을 적용한다. ASR의 동작 알고리즘은 그림 2이다.

그림 2의 진행과정에서 W는 32비트 임시 저장 변수이고, ASR의 위 첨자 i는 현재 상태를 나타내고 i+1은 다음 상태를 나타낸다.

$W = ASR^i_4$
$ASR^{i+1}_4 = (ASR^i_3 \gg 9)$
$ASR^{i+1}_3 = ((ASR^i_3 \ll 23) | (ASR^i_2 \gg 9)) \wedge W$
$ASR^{i+1}_2 = ((ASR^i_2 \ll 23) | (ASR^i_1 \gg 9)) \wedge W$
$ASR^{i+1}_1 = ((ASR^i_1 \ll 23) | (ASR^i_0 \gg 9)) \wedge (W \ll 2)$
$ASR^{i+1}_0 = (ASR^i_0 \ll 23) \wedge (W \ll 5) \wedge (W \ll 2) \wedge W$

그림 2. ASR 다음 상태 진행 알고리즘

3.3 축소함수

축소함수는 128비트 초기값을 가지고 S-박스와 같은 비선형 연산을 사용하지 않으면서 그림 1과 같이 151비트 ASR을 적용한 후 최종적으로 32비트 키를 생성한다. 먼저 32비트 초기값 AES^0_1과 AES^0_0를 b'()에 입력으로 적용하고, 그 결과 32비트를 ASR^1_4의 23비트 중 하위 5비트 값에 의한 데이터의존 회전연산(Data Dependent Rotation)을 수행한다. 그리고 그 결과 32비트와 ASR^1_1의 32비트를 b()에 적용한 후 SF()에 보낸다. 유사한 방법으로 AES^0_2과 AES^0_3을 적용하여 b()를 수행하며, 처음에 적용된 데이터의존 회전연산 값을 피하기 위해 ASR^1_4의 23비트 중 이번에는 상위 5 비트로 데이터의존 회전연산을 수행 후 SF()에 보낸다. AES^0_2는 아무런 변화 없이 SF()에 적용하며 마지막으로 ASR^1_0을 SF()에 적용한다. 이와 같이 새롭게 업 데이터 된 4개의 워드로 SF()를 수행 후 다시 업 데이터 된 4개의 워드 중 3번째 워드 AES^1_2와 ASR^1_3을 XOR연산 후 최종적인 AA32 스트림 암호의 출력으로 한다. 다음번 32비트 출력은 이와 같은 방법을 반복 적용한다.

축소함수의 비선형 연산을 수행하는 b(), b'(), SF()의 수행 알고리즘은 다음과 같다.

b()의 입력은 2개의 32비트 값을 입력으로 받는다. 입력 2개의 값은 각각 4개의 바이트 단위로 나누고 그림 3과 같이 바이트 단위의 AND, OR, 왼쪽 회전연산을 수행 후 다시 모든 바이트를 연결한 32비트 값을 만든다. b'()는 AND, OR 연산의 순서가 b()와 반대이다. 이는 AND와 OR 연산은 서로 쌍대성(Duality)을 가지며, b()와 b'() 사이에 데이터의존 회전 연산이 있지만 연속해서 같은 연산(AND, OR)의 반복은 비트 단위 암호분석에서 높은 확률의 어떤 비트를 생성할 가능성이 있다. 그래서 이를 피하기 위해 AND, OR 순서가 서로 다른 함수를 적용한다.

그리고 S-박스 대신 비선형변환을 위한 SF()는 4개

b(A, B), b`(A, B)
입력 A, B는 32비트 값
$A = a_3 \| a_2 \| a_1 \| a_0$
$B = b_3 \| b_2 \| b_1 \| b_0$

b()
$a_3 = (a_3 \,\&\, b_3) \ll b_3$
$a_2 = (a_2 \mid b_2) \ll b_2$
$a_1 = (a_1 \,\&\, b_1) \ll b_1$
$a_0 = (a_0 \mid b_0) \ll b_0$

b`()
$a_3 = (a_3 \mid b_3) \ll b_3$
$a_2 = (a_2 \,\&\, b_2) \ll b_2$
$a_1 = (a_1 \mid b_1) \ll b_1$
$a_0 = (a_0 \,\&\, b_0) \ll b_0$

출력 $A = a_3 \| a_2 \| a_1 \| a_0$ 한 32비트 값

그림 3. b(), b'() 진행 알고리즘

의 워드를 입력으로 받아 그림 4의 SF()를 수행 후 4개 워드 모두 새롭게 업 데이터 한다. 수행과정은 4개의 워드에서 각각 1비트 씩 4비트 입력을 받아 4비트 출력을 병렬로 수행한다.

```
SF(ASR^1_3, AES^0_2, AES^0_1, AES^0_0)

SF_3 = ASR^1_3,   SF_2 = AES^0_2,
SF_1 = AES^0_1,   SF_0 = AES^0_0
SF_2 ^= SF_0 | SF_1
SF_3 ^= ~(SF_1 & SF_2)
SF_0 ^= SF_2 | (SF_3 ^ SF_1)
SF_1 ^= SF_3 & SF_0
SF_2 ^= SF_3

AES^1_3 = SF_3,   AES^1_2 = SF_2,
AES^1_1 = SF_0,   AES^1_0 = SF_0

출력 AES^1_3, AES^1_2, AES^1_1, AES^1_0
```

그림 4. SF() 진행 알고리즘

Ⅳ. 구현결과

제안한 AA32의 소프트웨어 구현은 Visual Studio 2008 C 컴파일러를 사용하였고 실행환경은 Windows Vista, Intel Core(TM)2 Duo CPU 2.26Ghz, 2.27Ghz, 2GB RAM의 환경에서 알고리즘의 수행 시간을 테스트했다. 결과는 그림 5와 같다. 그림 5에서 각 알고리즘의 3Gb의 키 생성 시간을 막대그래프 안에 표시 했으며, 막대그래프 위의 정수는 AA32의 수행 시간을 100으로 환산한 각각의 수행 시간 비율값이다.

4.1 실험 결과 분석

그림 5의 분석을 위해 AA32와 비교된 각각의 알고리즘은 각 알고리즘의 개발자가 제공한 소스를 그대로 사용하였고 초기화과정은 제외하고 실제 키 생성 알고리즘만을 적용하여 얻은 결과이다. 그리고 전체 알고리즘을 10번 수행 후 최대치와 최저치를 제외한 8번의 결과값을 평균해서 반올림 했다. 스트림 암호는 생성된 키와 평문을 단순 XOR 연산 수행으로 암호문을 만들기 때문에 암호 복호 수행 시간보다는 키 생성 시간으로 스트림 암호의 수행 시간을 평가 한다.

AA32, SNOW, RC4, SSC2는 32비트 출력의 스트림 암호이고, Salsa20은 128비트 스트림 암호이다. 무선통신용 A5의 대안으로 개발된 SSC2보다 AA32가 약 67% 속도가 향상된 결과를 보여주고 있으며, 그 이유는

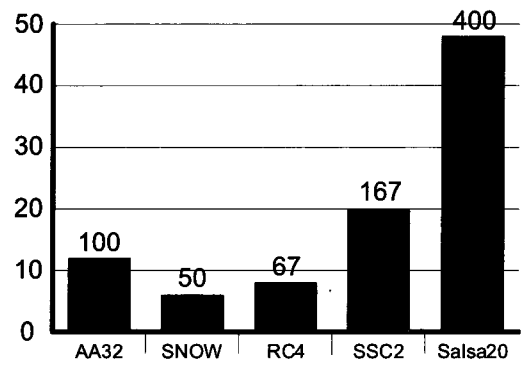


그림 5. 수행시간 테스트 결과

SSC2는 32비트 단위 LFSR이 4개, LFG(Lagged Fibonacci Generator)가 17개를 사용하여 32비트 출력 결과를 생성한다. AA32는 축소함수에 4개, ASR에 5개를 사용하여 SSC2보다 12개의 워드를 적게 사용하고 있다. 이와 같은 적은 메모리 사용이 실행 시간의 단축 결과를 보여 주었다.

그리고 SNOW, RC4는 AA32보다 빠르지만 두 알고리즘 모두 S-박스를 사용하고, 메모리 사용도 AA32보다 많다. 그래서 초경량의 하드웨어 환경에서 사용하기가 불가능하다.

그리고 Salsa20과의 비교는 ecrypt Ⅱ에서 주목받은 알고리즘으로 AA32와는 출력 크기가 다르지만 출력된 3Gb의 생성 키를 기준으로 실행 시간을 측정하였기 때문에 동일한 조건의 시간 테스트라고 말할 수 있다. 결과는 약 3배정도 AA32가 빠른 결과를 보여주고 있다.

4.2 안전성 분석

AA32의 안전성 분석은 그림 1에서 최종적인 32비트 출력인 AES^1_2의 분석 확률을 구해서 전체적인 AA32의 안전성을 입증한다. 그림 1에서 축소함수에 ASR은 상태변화에 따른 일종의 라운드 키 형태로 계속해서 적용되고 있다. 그래서 ASR의 전체적인 선형 복잡도 분석 확률을 먼저 하면, 참고문헌[9]에 의해 약 $n \leq LC \leq (n^2 + n)/2$ 이며, 기존의 LFSR은 선형 복잡도가 2n이다. n비트의 LFSR은 2n 개의 출력을 알고 있다면 초기값과 특성다항식을 풀이 할 수 있다. 그러나 ASR은 $(n^2 + n)/2$ 으로 기존의 LFSR 보다 동등하거나 큰 선형 복잡도를 가진다는 것이다. 그리고 ASR은 0을 제외한 최대 주기 수열을 생성한다. 이는 ASR의 특정 비트 또는 비트열(연속적이거나 연속적이지 않는 모든 경우)은 0을 제외한 모든 값들이 다음과 같이 동일한 출현 빈도수를 갖는다.

$$0 = \frac{2^{N-n}-1}{2^{N}-1}, \qquad 0 \neq \frac{2^{N-n}}{2^{N}-1} \tag{1}$$

수식 1에서 N은 최대주기수열의 비트 수이고 n은 특정 비트 또는 비트열의 수이다.

마지막으로 그림 6는 ASR 151비트의 다음 상태 진행을 그림으로 표현한 것이다.

두 번째 축소함수 수행에서 비선형 변환인 b(), b'()의 안전성 분석확률은 AND, OR, 데이터의존 회전연산의 분석을 통해 안전성을 입증할 수 있다. 먼저 논리연산(AND, OR)[26]의 안전성은 바이트 단위에서 1의 개수(1의 위치는 아무런 영향이 없음)에 따라 다음과 같은 분석확률을 구할 수 있다.

$$\frac{1}{2^{N}}\sum_{h=1}^{N}\binom{N}{h}\cdot 2^{-h} \tag{2}$$

수식 2에서 N은 바이트 단위 이므로 8이고, h는 바이트 단위에서 출현한 1의 개수이다. 수식 2를 계산하면 약 0.1이며, $2^{-3.25}$이다. 다음은 데이터의존 회전연산[27]의 안전성은 가변적인 회전 연산량 일 경우 전수조사 이외의 특별한 방법이 없다. 그래서 바이트 단위의 회전연산은 하위 3비트만 회전연산에 적용되는 값이므로 분석 확률은 2^{-3}이 된다. 그래서 최종적인 바이트 단위 분석 확률은 AND, OR 연산과 데이터의존 회전연산이 서로 독립이므로 곱한 값 $2^{-6.25} = 2^{-3.25} * 2^{-3}$이 되고, 각각의 바이트 단위 또한 독립적으로 적용되므로 전체 워드의 분석확률은 $2^{-25} = (2^{-6.25})^4$가 된다. 그리고 그림 1에서 b()와 b'() 사이에 워드 단위 데이터의존 회전연산이 독립적으로 적용되며 분석 확률은 2^{-5}이다. 그래서 SF()의 3번째 입력 워드의 최종적인 분석 확률은 $2^{-55} = (2^{-25})^2 * 2^{-5}$이 된다.

마지막으로 AA32의 또 다른 비선형 변환인 SF()의 3번째 입력 워드를 기준으로 분석해 보면, SF()는 4개의 입력 워드에서 각 워드의 동일한 위치의 4비트 입력으로 확산(Permutation)을 수행한 후 4비트 출력값을 동일한 위치의 각각의 워드로 보낸다. 그림 4의 알고리즘을 수행하면 그림 7과 같이 진행 된다.

예를 들어 SF()의 4개의 입력 워드에서 각 워드의 MSB가 "0000"이라면 SF()의 4개의 출력 워드의 MSB는 "1111"이 된다. 이와 같은 변환은 4비트 입력에 대해서 4비트 출력을 만드는 일종의 S-박스와 같은 역할을 SF()가 수행한다.

SF()의 4비트 입력에 대한 4비트 출력값은 다음과 같다.

Output = {0xF, 0x8, 0x7, 0x6, 0xB, 0xC, 0xE, 0xD,
0x0, 0x4, 0x9, 0xA, 0x5, 0x1, 0x3, 0x2}

그림 4의 알고리즘 진행으로 4비트 출력에 대한 최대 차분 및 선형 특성은 2^{-2}이고, 입력 비트에 대한 출력비트의 비선형 차수는 최대 3이다. 이는 SF()의 각각의 입력 비트에 대한 분석 확률로 한 워드일 경우는 $2^{-64} = (2^{-2})^{32}$이 된다. 그래서 최종적인 AA32의 분석 확률은 SF()의 3번째 입력 워드의 분석 확률 2^{-55}과 SF()의 수행 후 분석 확률 2^{-64}을 곱한 값인 2^{-119}이 된다. 그리고 최종 출력 32비트인 $AES^1{}_2$와 $ASR^1{}_3$을 XOR연산을 수행하여 표백(Whitening)하는 것은 AA32의 안전성에는 크게 영향을 미치지는 않지만 SF()의 세 번째 워드의 단순한 출력으로 SF() 입력 4개의 워드 분석에 사용될 수 있으며, 이는 AA32의 취약점이 될 수 있다. 이와 같이 SF()의 내부 상태를 숨기기 위해 XOR연산을 통한 표백과정을 수행하는 것이다.

제안한 AA32(2^{-119})의 안전성 분석을 통해 SSC2(2^{-52})보다 월등히 낮은 확률로 SSC2보다 높은 안전성을 보이고, 수행 속도 또한 약 67% 향상된 결과를 보여주고 있다.

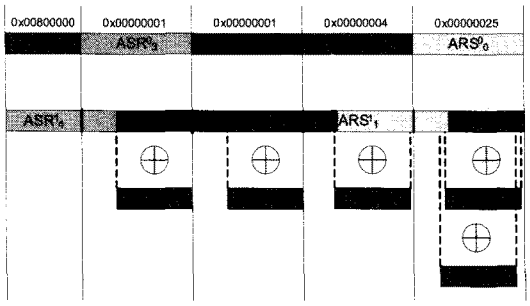


그림 6. ASR 진행 흐름도

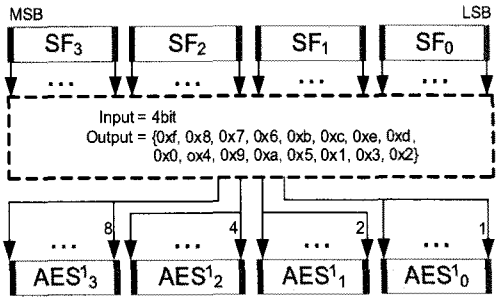


그림 7. SF 진행 흐름도

V. 결 론

안전하고 빠른 실시간처리가 가능한 32비트 출력의 스트림 암호 AA32를 제안하였다. AA32는 ASR 151비트와 128비트를 32비트로 축소시키는 비선형 변환 축소함수로 구성된 결합 함수 스트림 암호 알고리즘이다. ASR은 워드 단위 피드백을 수행 하므로 기존의 LFSR보다 수행 속도가 빠르며 안전성 또한 기존 LFSR보다 선형 복잡도가 높다. 그리고 비선형 변환 축소함수는 S-박스를 사용하지 않고 많은 계산량을 요구하는 산술 연산 대신 간단한 논리 연산만으로 구성하여 소프트웨어 및 하드웨어 구현 시 빠르게 수행 할 수 있다.

제안한 AA32는 SSC2, Salsa20 등과 수행 시간 테스트에서 최소 67%에서 최대 300%까지 수행 시간이 단축되었으며, 안전성 또한 SSC2보다 매우 높게 향상 되었다. 그리고 AA32는 휴대폰과 같은 무선 인터넷 환경과 DRM(Digital Right Management) 등과 같은 실시간처리가 필요한 분야와 제한된 환경인 무선 센서 네트워크(WSN)에 사용 가능한 고속 스트림 암호 알고리즘이다.

마지막으로 제안한 알고리즘을 하드웨어로 구현하고 128비트 출력의 스트림 암호 알고리즘으로 확장하여 더욱 빠른 Gbps급 스트림 암호 알고리즘개발을 다음 연구의 목표이다.

참 고 문 헌

[1] M. Blum and S. Micall, "How to generate cryptographically strong sequences of pseudo-random bits," *Procceedings of 25th IEEE Symposium on Foundations of Computer Science*, New York, pp.850-864, 1982.

[2] L. Blum, M. Blum and M. Shub, "A simple unpredictable pseudo-random number generator," *Siam J. on Computing*, pp.364-393, 1986.

[3] A. C. Yao, "Theory and applications of trapdoor functions," *Proceedings of the 25th IEEE Symposium on Foundations of Computer Science*, New York, pp.80-91, 1982.

[4] L. Kocarev, G. Jakimoski and Z. Tasev, "Choas and Pseudo-Randomness," *Choas Control: Theory and Applications, Lecture Notes in Control and Information Sciences*, Vol.292, pp.247-264, 2004.

[5] F. James, "A review of pseudo-random number generator," *Computer Physis Communications*, Vol. 60, pp.329-344, 1990.

[6] P. Ekdahl, T. Johansson, "SNOW-a new stream cipher," *NESSIE*, http://www.it.lth.se/cryptology/snow/

[7] FIPS PUB 197, *Advanced Encryption Standard (AES)*, NIST, 2001.

[8] D. J. Bernstein and P. Schwabe, "New AES Software Speed Records," *INDOCRYPT 2008*, LNCS Vol.5365, pp.322-336, 2008.

[9] 박창수, 조경연, "갈로이 선형 궤환 레지스터의 일반화.", *전자공학회논문지 제43권 CI편 제1호* 2006. 1.

[10] L. Brynielsson, "On the linear complexity of combined shift register sequences," *Advances in Cryptology-Eurocrypt '85* pp.156-166, 1986.

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

[12] D. J. Bernstein, Synchronous Stream Cipher Salsa20, http://www.ecrypt.eu.org/stream/salsa20.html

[13] "New European Schemes for Signatures. Integrity. and Encryption(NESSIE)" http://cryptonessie.org/.

[14] http://www.ecrypt.eu.org/

[15] M. Walker and T. Wright, "GSM and UMTS: The creation of global mobile communication," *John Wiley & Sons*, pp.385-406, 2002.

[16] B. Schneier, "Appled cryptography: Protocols, algorithms and source code in C," *John Wiley & Sons*, 1996.

[17] ETSI. 3GPP TS 35.201. Specification of the 3GPP Confidentiality and Integrity Algorithms: Document 1: f8 and f9 Specification, 2002.

[18] Bluetooth SIG. Specification of the Bluetooth System 2.0. http://www.bluetooth.com, 2005.

[19] A. Biryukov, A. Shamir and D. Wagner, "Real time cryptanalysis of A5/1 on a PC," *Proceedings of Fast Software Encryption-FES*, 2000.

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

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

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

[23] P. Souradyuti and B. Preneel, "Analysis of

Non-fortuitous RC4 key stream generator," *Progress in Crytology-INDOCRYPT*, 2003.

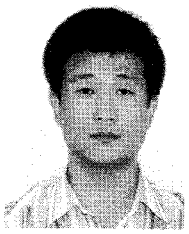
[24] L. of the IEEE CS, "Wireless LAN medium access control(MAC) and physical layer(PHY) specifications," *Technical Report, IEEE Standard 802.11*, 1999.

[25] E. Zenner, "Why IV Setup for Stream Cipher is Difficult," *Proceedings of Dagstuhl Seminar on Symmetric Cryptography*, 2007.

[26] Y.L. Yin, "A Note on the Block Cipher Camellia," *a contribution for ISO/IEC JTC1/SC27*, 2000.

[27] S. Contini, R. L. Rivest, M. J. B. Robshaw, and Y. L. Yin, "The Security of the RC6 Block Cipher," 1998.

김 길 호 (Gil-Ho Kim) 정회원



2000년 한국방송통신대학교 전자계산학과 학사

2002년 부경대학교 컴퓨터공학과 석사

2010년 부경대학교컴퓨터공학과 박사

<관심분야>반도체회로설계, 암호 알고리즘, 컴퓨터 구조

박 창 수 (Chang-Soo Park) 정회원

1995년 인제대학교 전자공학과 학사

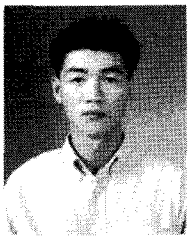
2001년 부경대학교 컴퓨터공학과 석사

2007년 부경대학교 컴퓨터공학과 박사

2008년~현재 부경대학교 누리사업단 계약교수

<관심분야>반도체회로설계, 암호 알고리즘, 컴퓨터 구조

김 종 남 (Jong-Nam Kim) 정회원



1995년 2월 금오공과대학교 전자공학과 학사

1997년 2월 광주과학기술원 정보통신공학과 석사

2001년 8월 광주과학기술원 기전공학과 박사

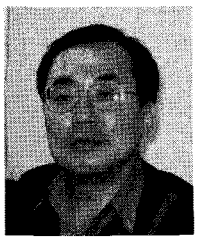
2001년 8월~2004년 2월 KBS 기술연구소 선임연구원

2004년 4월~현재 부경대학교 IT융합응용공학과 교수

2003년 3월~현재 (주)홈캐스트 사외이사

<관심분야> 영상신호처리, 멀티미디어 보안

조 경 연 (Jong-Nam Kim) 정회원



1990 인하대학교 공과대학전자공학과 정보공학 전공 박사

1983~1991 삼보컴퓨터 기술연구소 책임연구원

1991~현재 부경대학교 IT융합응용공학과 교수

1991~2001 삼보컴퓨터 기술연구소 비상임기술 고문

1998~현재 에이디칩스 사외이사 겸 비상임기술고문

<관심분야>전산기구조, 반도체회로설계, 암호 알고리즘