고속 스트림 암호 Canon

(A Fast stream cipher Canon)

김 길 호*
(Gil-Ho Kim)

요 약 기밀성(Confidentiality)과 무결성(Integrity)을 보장 할 수 있는 무선 센서 네트워크 구축에 필요한 스트림 암호 Canon을 제안한다. Canon은 128비트 비밀 키와 128비트 IV(Initial Vector)로 128비트 스트림 키를 생성하고, 생성된 키와 128비트 평문을 함께 표백(Whitening)처리를 통해 128비트 암호문을 만든다. 쉬운 하드웨어 구현과 빠른 소프트웨어 실행을 위해 Canon 알고리즘은 간단한 논리연산만으로 구성되어 있다. 특히 비선형(Non-Linear) 연산을 위한 S-박스를 사용하지 않기 때문에 하드웨어 구성이 매우 간결하다. 제안한 스트림 암호 Canon은 AES, Salsa20 보다 수행 속도 테스트결과 빠른 결과를 보여주고 있으며, Trivium보다 gate수가 작다. 그래서 Canon은 휴대폰과 같은 무선 인터넷 환경과 DRM(Digital Right Management)과 같은 실시간처리가 필요한 분야와 무선 센서 네트워크(Wireless Sensor Network), RFID 등과 같은 물리적 환경이 매우 제한적인 응용에 사용할 목적으로 소프트웨어 및 하드웨어 구현이 쉬운 128비트 스트림 암호이다.

핵심주제어 : 스트림 암호, LFSR, ASR, RFID, DRM, 무선 센서 네트워크

Abstract Propose stream cipher Canon that need in Wireless sensor network construction that can secure confidentiality and integrity. Create Canon 128 bits streams key by 128 bits secret key and 128 bits IV, and makes 128 bits cipher text through whitening processing with produced streams key and 128 bits plaintext together. Canon for easy hardware implementation and software running fast algorithm consists only of simple logic operations. In particular, because it does not use S-boxes for non-linear operations, hardware implementation is very easy. Proposed stream cipher Canon shows fast speed test results performed better than AES, Salsa20, and gate number is small than Trivium. Canon purpose of the physical environment is very limited applications, mobile phones, wireless Internet environment, DRM (Digital Right Management), wireless sensor networks, RFID, and use software and hardware implementation easy 128 bits stream ciphers.

Key Words : Stream Cipher, LFSR, ASR, RFID, DRM, wireless sensor network

1. 서 론

다수의 센서노드로 구성된 무선 센서 네트워크는 주변의 사람, 사물 그리고 환경정보를 인식하여 현재 상황을 판단한다. 이 과정에서 각 센서 노드에서 처리되는 정보는 신뢰성이 매우 중요하며, 특히 각 센서노드는 물리적인 제한 때문에 데이터의 저장, 계산 능력, 배터리, 대역폭 등에 분명한 한계를 보이고 있다. 그래서 신뢰성과 안전성이 확보된 센서 네트워크 구축을 위해 추가적으로 정보의 저장 및 전송에 암호의 적용이 필요하며, 암호 알고리즘의 구현은 센서 노드에서 한정된 자원을 어떻게 효율적으로 활용할 것인가에 관한 연구가 반드시 필요하다.

* 부경대학교 IT융합응용공학과

현재 센서노드 상에서 구현된 암호 알고리즘은 공개 키 암호[1]와 블록 암호인 AES[2,3]가 있으나, 이들은 계산 량과 계산에 필요한 메모리도 많이 필요로 한다. 이는 센서노드의 물리적 한계점과 정면으로 반대되는 것으로 이를 극복할 대안으로 최근에 3단계로 진행된 eSTREAM[4,5] 프로젝트에서 소프트웨어와 하드웨어 부문에서 몇 개의 스트림 암호가 선택되었다. 센서노드에서 스트림 암호의 적용은 공개 키 암호와 블록 암호 알고리즘보다 가볍기 때문에 구현이 쉬운 장점도 있지만 스트림 암화만으로는 기밀성과 무결성의 완전한 보장이 되지 않는 단점도 있다.

예를 들어 센서 노드에서 무선으로 송수신 하는 정보는 데이터의 양이 매우 작다. 그리고 평문과 단순히 스트림 키의 XOR연산은 생성된 암호문과 일대일 대응되므로 공격자는 암호문 전체를 공격하지 않고 특정부분의 바이트, 또는 임의의 비트를 어떤 특정 값으로 변경함으로써 공격이 쉽게 이루어질 수 있다. 이를 방지하기 위해 블록 암호처럼 운영모드를 적용할 수 있지만, 센서 노드 상에서 송수신 되는 데이터의 양이 작기 때문에 모드의 적용이 쉽지 않고 그리고 모드 적용을 위해서는 이전에 생성된 암호문이나 IV값을 메모리에 저장하고 있어야 하므로 추가적인 메모리가 필요해진다. 이는 물리적 환경이 매우 제한적인 무선 센서노드 구현을 어렵게 한다.

이에 본 논문에서는 기밀성과 무결성을 보장하면서 휴대폰과 같은 무선 인터넷 환경과 DRM과 같은 실시간처리가 필요한 분야와 무선 센서 네트워크, RFID 등과 같은 물리적 환경이 매우 제한적인 응용에 사용할 목적으로 소프트웨어 및 하드웨어 구현이 쉬운 128비트 스트림 암호 Canon을 제안한다. Canon은 참고문헌 [6]을 확장 및 최적화한 것으로 128비트 비밀 키와 128비트 IV를 사용하여 128비트 암호문을 만드는 스트림 암호 알고리즘으로 수행속도 향상 및 안전성이 매우 강화되었다.

Canon의 구성은 277비트 산술 쉬프트 레지스터(Arithmetic Shift Register)[7]와 비선형 변환과 확산으로 구성된 혼잡함수(Confusion function)를 통해 128비트 스트림 키를 생성하고 생성된 키와 평문 128비트는 단순히 XOR연산을 수행하지 않고, 기밀성과 무결성 보장을 위해 표백 처리과정을 수행하여 128비트 암호문을 만든다. 그리고 Canon에서 사용된 모든 연산은 간단한 논리 연산으로만 구성되어 있어 소프트웨어 및 하드웨어 구현이 용이하다. 특히 비선형 변환 연산은 S-박스나 계산 복잡도가 높은 곱셈, 나눗셈, 지수, 유한체 연산을 사용하지 않아 제한적인 하드웨어인 센서노드에 쉽게 적용할 수 있는 암호 알고리즘이다.

제안한 Canon은 AES, Salsa20[8] 보다 소프트웨어 수행 속도 테스트결과 최소 15%에서 최대 50% 정도 빠른 결과를 보여주고 있으며, 안전성 또한 만족하고 있으며 하드웨어 구현 역시 설명한 어플리케이션에 필요한 속도를 충분히 만족하고 있다.

본 논문의 구성은 2장에서 무선 센서 네트워크를 위해 개발된 여러 암호 알고리즘과 스트림 암호 개발 프로젝트에서 발표된 알고리즘의 간략한 소개와 문제점을 중심으로 설명하고, 3장에서 제안한 Canon 알고리즘을 자세히 설명하고, 4장에서 Canon의 소프트웨어, 하드웨어 구현 결과 및 안전성에 대해 분석하고, 마지막으로 결론으로 끝맺는다.

2. 관련연구

비밀 키 암호를 사용하여 센서노드 통신에 널리 사용되고 있는 TI사의 CC2420[2], CC2430은 AES-128 암호 알고리즘을 하드웨어 가속기로 제공 하고 있으며, 운영모드로 Counter, CBC-MAC, CCM모드를 지원하고 있다. 현재까지 속도는 무선 센서 네트워크상에서 충분히 사용할 수 있지만 경제적인 측면에서 더 이상 센서노드의 단가를 낮출 수 없는 것이 문제이다. 그리고 센서노드에서 공개 키 암호의 적용은 128, 160, 192비트 타원 곡선 암호 알고리즘(ECC)을 TinyOS 상에서 암호화, 키 분배, 전자 서명 프로토콜을 구현하였다[1]. 현재는 전자 서명에 3.17초 검증에 4.04초가 소요되어 실제로 응용에 사용될 수 있지만 차후 복잡한 WSN인 경우에는 속도가 문제될 수 있다. 그리고 공개 키 암호는 비밀 키 암호보다 전력 소비가 많아 각 센서 노드의 수명에도 많은 영향을 미친다.

현재 WSN 상에서 구현된 스트림 암호는 없지만 블록 암호와 공개 키 암호에 비해 가볍기 때문에 WSN에서 스트림 암호를 적용하기 위한 연구가 많이 진행되고 있고, 스트림 암호 개발을 위한 국제적인 공모사업으로 NESSIE(New European Schemes for Signatures, Integrity, and Encryption)[9], eSTREAM

이 있었으며 NESSIE 프로젝트에서 제안된 스트림 암호는 최종 평가에서 수행속도, 동기화 문제 등 여러 가지 면에서 만족할 결과를 얻지 못해 최종적으로 선택된 알고리즘 없이 종료하였고, 다음으로 진행된 eSTREAM에서는 3차에 걸쳐 소프트웨어[10] 및 하드웨어[11] 구현 부문으로 나누어 진행된 분석 및 평가의 결과로, HC-128, Rabbit, Salsa20, SOSEMANUK, Grain, MICKEY, Trivium 암호를 선택했다[12]. 현재까지 진행된 국제공모사업에서 스트림 암호가 선정되지 못한 주된 원인은 스트림 암호의 수행 속도가 블록 암호보다 월등히 앞서지 못했다는 것이다. 이는 LFSR 기반의 스트림 암호의 한계를 보여주는 것이다. 그래서 앞으로 스트림 암호의 개발 방향은 Gbps급 초고속 스트림 암호나 초경량 하드웨어 환경에 맞는 스트림 암호의 개발이 절실히 요구되고 있다.

다음은 본 논문에서 Canon과 수행 테스트 비교를 위한 알고리즘들을 간략히 소개한다.

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 키 수열과 랜덤함수를 구별하는 distinguishing 공격[13]과 WEP에 적용되는 경우 재동기 과정[14]에서 문제점이 발견되었다. 그리고 RC4는 256바이트의 메모리를 사용하므로 초경량 하드웨어 환경에서는 적합하지 않다.

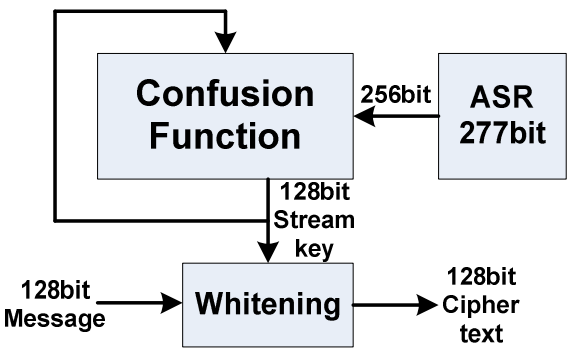
Salsa20은 eSTREAM 프로젝트에서 소프트웨어 구현 부문에 3차까지 주목받은 알고리즘으로 128비트 키와 64비트 IV를 사용해 키 스트림을 생성한다. 키 생성을 위한 내부 상태공간은 32비트로 된 4 X 4 테이블로 총 64바이트이다. 내부 상태의 갱신은 64바이트 입력과 64바이트 출력을 위한 해시 함수를 통해 이루어지고 이러한 해시 함수는 여러 개의 quarterround로 구성 되어 있으며, 각 quarterround는 XOR, 덧셈, 회전 연산으로 구성되어 있다.

Trivium[15] 역시 eSTREAM의 하드웨어 구현에서 최종 선택된 알고리즘으로 80비트 키와 80비트 IV를 사용하며, LFSR을 기반으로 288비트 내부 상태 변환을 통해 스트림 키를 생성하는 알고리즘이다. 하드웨어 구현 시 키 생성 알고리즘을 병렬로 구현하여 수행 속도가 매우 빠르고 전력 소비 또한 매우 효율적이다.

3. Canon

Canon 알고리즘을 설명하기 위해 사용된 그림이나 수식 등에서 논리 및 산술 연산자들은 일반적인 표기법을 그대로 적용하고, 대문자로 표기된 변수는 32비트이며, 소문자는 8비트이다. 그리고 변수 명에 위첨자는 상태 값을 나타내며, 아래첨자는 워드 또는 바이트 단위 순서 번호이다.

<그림 1>은 Canon의 128비트 암호문을 생성하는 전체 진행과정을 보여주고 있다. Canon은 128비트 비밀 키와 128비트 IV를 사용하여 128비트 스트림 키를 생성한다. 그리고 모든 스트림 암호는 생성된 스트림 키와 평문이 단순히 XOR연산을 통해 암호문을 만들지만, Canon은 표백과정을 적용하여 암호문을 만드는 매우 독특한 특징이 있다. <그림 1>에서 277비트의 ASR을 새로 갱신한 후 256비트를 혼잡함수 수행을 위해 보내고 혼잡함수는 128비트 스트림 키를 생성한다. 생성된 128비트 스트림 키는 128비트 평문과 함께 표백처리 과정을 통해 최종적인 128비트 암호문을 생성한다. 이어지는 다음 절부터 <그림 1>의 순서대로 진행과정을 자세히 설명한다.



<그림 1> Canon 블록도

3.1 초기화 과정

초기화 과정은 ASR 277비트(32비트 8개와 마지막 최상위 워드는 21비트만 사용)와 혼잡함수 128비트의

초기상태를 만드는 과정으로 스트림 암호에서는 최종적인 키 생성 과정에서 초기상태는 매우 중요하며[16], 초기상태 값의 노출은 스트림 암호의 안전성과 밀접한 관계가 있다. 초기상태 값을 만들기 위해 사용자 비밀 키와 IV를 암호학적으로 안전한 의사난수함수에 입력하여 필요한 405비트(277+128)를 초기화시켜야 한다. 초기화 과정은 다음과 같다.

먼저 IV 128비트를 ASR의 ASR_0, ASR_2, ASR_5, ($ASR_8 \| ASR_7$)에 대입하고 특히 IV의 최상위 32비트는 ASR_8의 21비트와 ASR_7의 상위 11비트이다. 나머지 ASR들은 모두 1로 채운 후 1024번의 ASR 갱신과정을 수행한다. 유사한 방법으로 비밀 키 128비트는 혼잡함수 128비트에 순서대로 대입한 후 혼잡함수를 1024번 반복 수행하여 갱신된 값을 혼잡함수의 초기값이 되며 혼잡함수 수행은 ASR과 병행하므로 ASR은 2048번 째 갱신된 값이 초기 값이 된다.

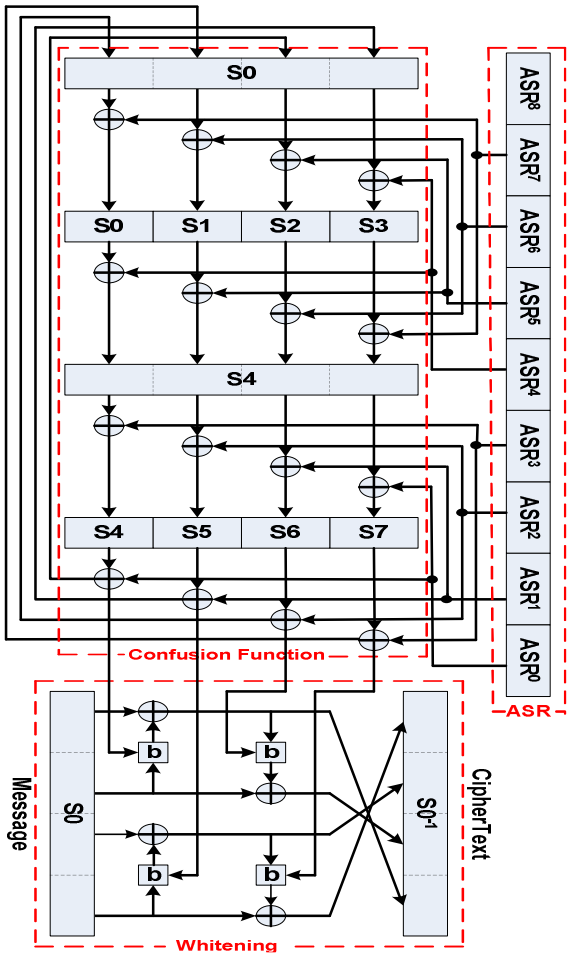
3.2 ASR

산술 쉬프트 레지스터(ASR)는 $GF(2^n)$상에서 0이 아닌 초기 값에 0 또는 1이 아닌 임의의 수 D를 곱하는 수열로 정의한다. ASR의 i번째 값(상태) ASR^i는 $ASR^0 \cdot D^i$가 된다. $D^k = 1$이 되는 t가 $t = 2^{n-1}$로 유일하게 되는 기약 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 복잡도보다 높아서 안전도가 높고, 다음 상태 갱신을 위한 연산이 32비트 단위로 처리되므로 수행속도도 기존의 LFSR보다 빠르다.

$W=ASR^i_8$
$ASR^{i+1}_8=(ASR^i_7 \gg 11)$
$ASR^{i+1}_7=((ASR^i_7 \ll 21)|(ASR^i_6 \gg 11))\hat{}W$
$ASR^{i+1}_6=((ASR^i_6 \ll 21)|(ASR^i_5 \gg 11))\hat{}W$
$ASR^{i+1}_5=((ASR^i_5 \ll 21)|(ASR^i_4 \gg 11))\hat{}W$
$ASR^{i+1}_4=((ASR^i_4 \ll 21)|(ASR^i_3 \gg 11))\hat{}W$
$ASR^{i+1}_3=((ASR^i_3 \ll 21)|(ASR^i_2 \gg 11))\hat{}W$
$ASR^{i+1}_2=((ASR^i_2 \ll 21)|(ASR^i_1 \gg 11))\hat{}W$
$ASR^{i+1}_1=((ASR^i_1 \ll 21)|(ASR^i_0 \gg 11))\hat{}(W \ll 1)$
$ASR^{i+1}_0=(ASR^i_0 \ll 21)\hat{}(W \ll 4)\hat{}(W \ll 2)\hat{}W$

<그림 2> ASR 다음 상태 알고리즘

Canon에서는 $GF(2^{277})$상에서 특성다항식은 16진수로 0x00200000 0x00000001 0x00000001 0x00000001 0x00000001 0x00000001 0x00000001 0x00000002 0x00000015, $D = 2^{21}$을 적용한다. ASR의 동작 알고리즘은 <그림 2>이다. <그림 2>의 진행과정에서 W는 32비트 임시 저장 변수이고, ASR의 위 첨자 i는 현재 상태를 나타내고 i+1은 다음 상태를 나타낸다.



<그림 3> Canon 흐름도

3.3 혼잡함수

혼잡함수는 초기 값 128비트와 ASR로부터 입력 받은 256비트를 사용하여 최종적인 128비트 스트림 키를 생성하는 함수이다. 진행과정은 <그림 3>의 Confusion Function 박스 내의 흐름을 보인다. 혼잡함수는 크게 2단계로 나눌 수 있고 첫 번째 단계는 초기 값 128비트를 사용하여 S0함수를 수행하고 ASR로

부터 입력 받은 256비트 중 상위 128비트(ASR_7, ASR_6, ASR_5, ASR_4)와 낮은 위치 순서대로 XOR연산을 수행 한다. 다음은 128비트를 32비트 단위로 나누어 각각 S0, S1, S2, S3함수를 수행하고 앞서 적용한 ASR 128비트를 역순으로 XOR연산을 수행한다. 두 번째 단계는 첫 번째 단계의 출력 결과 32비트 4개를 다시 128비트로 합친 후 S4함수를 수행한다. 이번에는 ASR의 나머지 128비트(ASR_3, ASR_2, ASR_1, ASR_0)를 낮은 위치 순서대로 XOR연산을 수행 한다. 그리고 128비트를 다시 32비트 단위로 나누어 각각 S4, S5, S6, S7함수를 수행하고 앞서 적용한 ASR 128비트를 역순으로 XOR연산을 수행하여 최종적인 Canon 암호의 128비트 스트림 키를 생성한다. 생성된 128비트 키는 Whitening함수에 적용됨과 동시에 다음 128비트 스트림 키 생성을 위해 64비트 단위로 치환(Permutation)하여 S0함수의 입력으로 된다.

3.3.1 128비트 S0, S4함수

혼잡함수 내의 비선형 연산을 수행하는 128비트 처리함수는 S0과 S4함수가 있다. S-박스 대신 비선형변환을 위한 S0과 S4함수는 입력 128비트를 내부 처리 과정에서 4개의 32비트로 나누어 수행 후 결과는 4개의 32비트 총 128비트를 만든다. <그림 4>가 S0, S4 함수의 수행 알고리즘이다.

```
S0(PT3, PT2, PT1, PT0)
  PT2 ^= PT0 | PT1
  PT3 ^= ~(PT1 & PT2)
  PT0 ^= PT2 | (PT3 ^ PT1)
  PT1 ^= PT3 & PT0
  PT2 ^= PT3

S4(PT3, PT2, PT1, PT0)
  PT2 ^= PT0 & PT1
  PT3 ^= PT1 | (PT2 ^ PT0)
  PT0 ^= PT2 & PT3
  PT1 ^= ~PT3 | PT0
  PT2 ^= PT3
```

<그림 4> S0, S4 알고리즘

3.3.2 32비트 S1, S2, S3, S5, S6, S7함수

혼잡함수 내의 비선형 연산을 수행하는 32비트 처리함수는 S0 ~ S7함수가 있다. 이 함수들 역시 S-박스 대신 비선형변환을 한다. 32비트 처리용 혼잡함수들은 32비트 입력을 내부에서 8비트 단위로 나누어 처리하여 4개의 8비트 총 32비트 결과를 만든다. <그림 5>는 32비트 혼잡함수의 수행 알고리즘이다. 여기서 빠진 S0과 S4함수는 <그림 4>의 알고리즘과 같고 내부적으로 처리하는 단위만 32비트에서 8비트로 전환하면 된다. 비선형변환(S0 ~ S7)은 4개의 입력 바이트 또는 워드에서 각 바이트 또는 워드의 동일한 위치의 4비트 입력으로 확산을 수행한 후 4비트 출력 값을 동일한 위치의 각각의 바이트 또는 워드로 보낸다. 이는 각각의 바이트 또는 워드에 bit-slice 모드를

```
S1(pt3, pt2, pt1, pt0)
  pt2 ^= pt0 | pt1
  pt3 ^= pt1 & pt2
  pt0 ^= (pt2 ^ pt1) | pt3
  pt1 ^= ~(pt3 & pt0)
  pt2 ^= pt3

S2(pt3, pt2, pt1, pt0)
  pt2 ^= pt0 & pt1
  pt3 ^= pt1 | pt2
  pt0 ^= ~(pt3 & pt2)
  pt1 ^= pt3 | (pt0 ^ pt2)
  pt2 ^= pt3

S3(pt3, pt2, pt1, pt0)
  pt2 ^= pt0 & pt1
  pt3 ^= ~(pt1 | pt2)
  pt0 ^= pt2 & pt3
  pt1 ^= (pt3 ^ pt2) | pt0
  pt2 ^= pt3

S5(pt3, pt2, pt1, pt0)
  pt2 ^= ~(pt0 & pt1)
  pt3 ^= pt1 | (pt0 ^ pt2)
  pt0 ^= pt2 & pt3
  pt1 ^= pt3 | pt0
  pt2 ^= pt3

S6(pt3, pt2, pt1, pt0)
  pt2 ^= pt0 & pt1
  pt3 ^= (pt1 ^ pt0) | pt2
  pt0 ^= pt2 & pt3
  pt1 ^= pt3 | ~pt0
  pt2 ^= pt3

S7(pt3, pt2, pt1, pt0)
  pt2 ^= pt0 & pt1
  pt3 ^= (pt1 ^ pt0) | ~pt2
  pt0 ^= pt2 & pt3
  pt1 ^= pt3 | pt0
  pt2 ^= pt3
```

<그림 5> S1, S2, S3, S5, S6, S7 알고리즘

적용한 것과 같아서 비선형 변환과 치환 효과를 동시에 볼 수 있다. 예를 들어 S0의 4개의 입력 워드에서 각 워드의 MSB가 “0000”이라면 S0함수의 4개의 출력 워드의 MSB는 “1111”이 된다. 이와 같은 변환은 4비트 입력에 대해서 4비트 출력을 만드는 일종의 S-박스와 같은 역할을 혼잡함수의 비선형변환 함수들이 수행한다고 할 수 있다.

3.4 표백

최종적인 128비트 암호문 생성은 혼잡함수를 통해 생성된 128비트 스트림 키와 입력된 128비트 평문을 표백처리 과정을 거쳐 128비트 암호문이 생성된다. <그림 3>의 맨 밑에 있는 Whitening 박스가 표백의 진행과정이다. 입력된 128비트 평문은 앞서 설명한 S0 함수를 수행한 후 32비트 단위로 나누어 혼잡함수에서 생성된 키와 b함수를 수행한다. b함수는 2개의 32비트 값을 입력 받아 바이트 단위로 나누어 AND, OR연산을 번갈아 수행 후 결과를 32비트로 만든다. 그리고 마지막 단계로 $S0^{-1}$함수를 수행한 결과가 최종적인 암호문이 된다. 여기서 $S0^{-1}$함수는 S0함수의 역변환을 수행하는 함수이다. 자세한 b함수와 $S0^{-1}$함수는 <그림 6>에 보인다.

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

$S0^{-1}(PT_3, PT_2, PT_1, PT_0)$
$PT_2 \mathbin{\hat{}}= PT_3$
$PT_1 \mathbin{\hat{}}= PT_3 \,\&\, PT_0$
$PT_0 \mathbin{\hat{}}= PT_2 \mid (PT_3 \mathbin{\hat{}} PT_1)$
$PT_3 \mathbin{\hat{}}= \sim(PT_1 \,\&\, PT_2)$
$PT_2 \mathbin{\hat{}}= PT_0 \mid PT_1$

<그림 6> b, $S0^{-1}$ 알고리즘

표백처리는 입력된 128비트 평문과 128비트 암호문이 일대일 대응관계를 없애기 위해 128비트 단위로 확산 시킬 수 있는 S0, $S0^{-1}$함수를 사용하였고, 64비트 단위 확산은 b함수, 마지막으로 32비트 단위 확산은 <그림 3>에서와 같이 $S0^{-1}$함수의 입력과 같이 처리한다. 이와 같은 표백처리는 평문과 암호문 사이에 암호키를 단순히 XOR연산만 수행하는 다른 스트림 암호보다 연관성을 없앨 수 있고 혼잡함수를 통해 생성된 128비트 스트림 키를 완전히 숨기는 기능도 표백에서 담당하고 있다.

마지막으로 Canon 스트림 암호의 복호는 지금까지 설명한 알고리즘을 그대로 적용하면 되고 단지 표백처리 과정에서 적용된 혼잡함수의 128비트 스트림 키를 역순으로 적용하면 된다. 그 이유는 표백처리 과정이 블록 암호의 1라운드 SPS^{-1}구조[17]를 이루고 있기 때문이다.

4. Canon의 성능 및 안전성 분석

제안한 Canon의 소프트웨어 구현은 Visual Studio 2010 C++ 컴파일러를 사용하였고 실행환경은 Windows7, Intel Core2 Duo CPU 2.26GHz, 2.27GHz, 2GB RAM의 환경에서 알고리즘의 수행 시간을 테스트했다. 결과는 <표 1>과 같다. <표 1>에서 각 알고리즘은 약 12Gb의 키 생성 시간과 암호문 생성 시간(괄호 안의 시간)을 2번째 열에 표시했으며, 마지막 열은 각 알고리즘들의 소프트웨어 수행 중 내부 상태 변환을 위한 전역 변수 메모리양을 비교한 값이다. 이 메모리는 하드웨어 구현 시 반드시 필요한 메모리와 거의 일치한다.

<표 1> 12Gb 스트림 키 생성시간과 사용된 메모리 비교

알고리즘	수행시간	메모리 사용
RC4	32(34) sec	256 Byte
salsa20[8]	46(48) sec	64 Byte
AES[3]	37(37) sec	1024 Byte
Canon	22(32) sec	67 Byte

4.1 소프트웨어 분석

<표 1>의 분석을 위해 Canon과 비교된 각각의 알고리즘은 각 알고리즘의 개발자가 제공한 소스를 그대로 사용하였다.

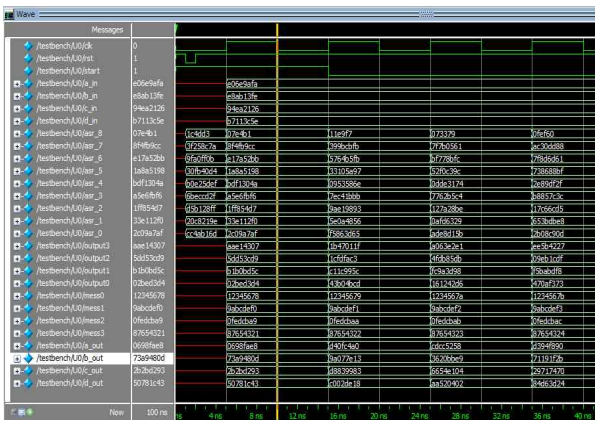
RC4는 32비트 출력의 스트림 암호이고, 나머지 알

고리즘들은 128비트 스트림 암호이다. 비교된 모든 알고리즘보다 Canon은 소프트웨어 수행 속도에서 앞서고 있다. 그리고 RC4, AES는 내부 상태 변환에 S-박스를 사용하여 메모리 사용이 Canon보다 훨씬 많다. 따라서 초경량의 제한적인 하드웨어 환경에서 사용하기가 불가능하다.

그리고 AES와 Salsa20과의 비교는 AES의 경우 eSTREAM에서 제안된 스트림 암호의 성능 평가의 기준이었으며, Salsa20은 eSTREAM 3차까지 소프트웨어 구현 부문에서 가장 주목받은 알고리즘이다. 결과는 AES보다는 약 15%, Salsa20보다는 약 50% 이상 Canon이 빠른 결과를 보여 주고 있다. 이는 적은 메모리 사용과 간단한 논리 연산만으로 구성된 알고리즘이 실행 시간을 단축하는 결과를 보인다.

4.2 하드웨어 구현

하드웨어 구현은 Modelsim 6.5d를 활용하여 제안한 알고리즘의 기능을 검증하였고 성능은 Quartus II 12.0을 활용하여 분석한 결과, Altera Cyclone II FPGA에서 Compile한 결과 Total Logic Elements : 1860 / 33216(6%), Total Registers : 533, Total Pins: 387 / 475, 이며, Trivium 64비트에서 NAND gate수는 5504[15]로 Trivium보다 Canon의 gate수가 작다. 그리고 Worse Case에서 Max Frequency는 113.28MHz (14.5Gbps)의 매우 빠른 성능을 보여주었다. 이는 무선 인터넷과 센서 네트워크 및 DRM 환경의 속도를



<그림 7> 하드웨어 시뮬레이션 결과

충분히 만족함을 보여준다. <그림 7>은 Modelsim을 통한 Canon의 수행을 시뮬레이션한 결과로 Canon의 내부 상태 변환 및 128비트 암호문 출력 스트림이 정확히 생성되는 것을 보여주고 있다.

이와 같은 결과는 실제 센서노드에서 성능과 다소 차이가 있을 것으로 예상되며 차후 연구에서는 시뮬레이션 검정이 아닌 실체 센서노드에서 실행된 결과를 반영하도록 하겠습니다.

4.3 안전성 분석

Canon의 안전성 분석은 <그림 3>의 혼잡함수의 128비트 출력 분석 확률을 구해서 전체적인 Canon의 안전성을 입증한다. <그림 3>에서 혼잡함수에 ASR은 상태변화에 따른 일종의 라운드 키 형태로 계속해서 적용되고 있다. 따라서 ASR의 전체적인 선형 복잡도 분석 확률을 먼저 구하면, ASR의 최소 선형 복잡도는 277비트이다. 즉 554개 출력 값을 알면 ASR의 모든 상태를 분석할 수 있다. 1/554의 확률은 약 2^{-9}이다. 이는 ASR의 안전성이 약 2^{-9}이라고 추정할 수 있다. 그리고 ASR은 0을 제외한 최대 주기 수열을 생성한다. 이는 ASR의 특정 비트 또는 비트열(연속적이거나 연속적이지 않는 모든 경우)은 0을 제외한 모든 값들이 동일한 출현 빈도수를 보여줌으로 특정한 값을 추정한 어떤 공격[18]에도 내성이 있다.

그리고 혼잡함수의 안전성은 비선형 변환인 S0 ~ S7함수의 안전성 분석을 통해 구할 수 있다. S0 ~ S7 함수의 안전성은 4개의 입력 바이트 또는 워드에서 각 바이트 또는 워드의 동일한 위치의 4비트 입력으로 확산을 수행한 후 4비트 출력 값을 동일한 위치의 각각의 바이트 또는 워드로 보낸다.

S0: f, 8, 7, 6, b, c, e, d, 0, 4, 9, a, 5, 1, 3, 2
S1: 2, 6, b, 8, 7, 3, 1, 0, d, a, 4, 5, 9, e, f, c
S2: 3, 0, d, 9, a, b, 8, c, f, e, 1, 4, 5, 6, 7, 2
S3: e, f, 2, 5, 6, 7, 4, 1, 0, 3, c, a, b, 8, 9, d
S4: 2, f, e, a, b, 7, 9, d, c, 3, 0, 5, 6, 8, 4, 1
S5: b, 7, 9, d, 0, f, c, 8, 4, a, 6, 1, e, 3, 2, 5
S6: 2, f, c, 8, b, a, 9, 3, e, 1, 0, 7, 6, 5, 4, d
S7: e, f, c, 5, 4, a, 9, d, 0, 3, 2, 8, b, 7, 6, 1

<그림 8> S0 ~ S7의 4비트 출력

<그림 8>은 비선형변환(S0 ~ S7)의 4비트 입력(0

~ f)에 대한 출력을 표시한 것이다. 4비트 출력에 대한 최대 차분 및 선형 특성은 2^{-2}이고, 입력 비트에 대한 출력비트의 비선형 차수는 최대 3이다. 이는 비선형변환의 전체 입력 비트에 대한 분석 확률이 128비트 입력일 경우는 2^{-64}가 된다. 이와 같은 과정이 4번 수행하므로 분석 확률은 2^{-256}이 된다. 따라서 최종적인 Canon의 분석 확률은 혼잡함수 분석 확률 2^{-256}과 ASR의 분석 확률 2^{-9}를 곱한 값인 2^{-265}가 된다. 그리고 마지막 표백처리는 Canon의 안전성에는 크게 영향을 미치지는 않지만 평문과 암호문 사이의 연관성을 없애면서 생성된 스트림 키의 노출을 피하여 Canon 내부 상태 분석에 어려움을 주기 위해 표백처리 과정을 수행하는 것이다.

계산적인 안전성으로 위와 같이 안전성이 입증되었지만, 상관관계 분석(Correlation Analysis)[19]과 대수적 분석(Algebraic Analysis)[20]에 대한 안전성은 입증하지 못했다. 그러나 제시한 알고리즘 내부의 13개 워드들이 비선형으로 확장 혼합 갱신되므로 ASR과의 상관관계를 이용하여 키를 찾는 상관 공격에 강하고, 갱신된 모든 비트가 출력에 영향을 미침으로 변수가 많아, 대수방정식을 이용하여 키를 복구하는 대수공격에도 강하다고 보이며, 특히 혼잡함수 내부 비선형변환은 S-박스를 사용하지 않으므로 Timing 공격[21]에 강하고, bit-slice형태로 비선형변환을 구성하여 Square 공격[22]에도 매우 강할 것으로 여겨진다.

5. 결 론

안전하고 빠른 하드웨어 구현 및 실시간처리가 가능한 128비트 출력의 스트림 암호 Canon을 제안하였다. Canon은 ASR 277비트, 혼잡함수 128비트와 표백으로 구성하여 최종적인 128비트 암호문을 만드는 스트림 암호 알고리즘이다. 혼잡함수 내의 비선형변환을 위한 S0 ~ S7함수는 S-박스를 사용하지 않고 또 많은 계산 량을 요구하는 유한체 연산 대신 간단한 논리 연산만으로 구성하여 소프트웨어 및 하드웨어 구현에서 쉽고 빠르게 수행 되도록 설계하였다.

제안한 Canon은 AES, Salsa20 등과 소프트웨어 수행 시간 테스트에서 최소 15.6%에서 최대 50%까지 수행 시간이 단축되었으며, 하드웨어 구현 시 사용되는 메모리 또한 매우 작다. 따라서 Canon은 휴대폰과 같은 무선 인터넷 환경과 DRM 등과 같은 실시간처리가 필요한 분야와 제한된 자원 환경인 무선 센서 네트워크의 단말 센서노드, RFID tag에 활용 가능한 고속 스트림 암호 알고리즘으로 평가된다.

추후 연구 과제로 제안한 알고리즘을 좀 더 확장 개선하여 더욱 고속인 Gbps급 스트림 암호 알고리즘을 개발함으로써 실시간으로 고속 이동 무선 통신 환경의 암호화에 적용하고자 한다.

참 고 문 헌

[1] “TinyECC: A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks,” Ver 1.0, http://discovery.csc.ncsu.edu/software/TinyECC/, 2007.

[2] CC2420 DataSheet, "CC2420 2.4GHz IEEE 802.15.4/ZigBee-ready RF Transceiver," Chip-con, 2006.

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

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

[5] http://www.ecrypt.eu.org/stream/phase3list.

[6] 김길호, 박창수, 김종남, 조경연, “소프트웨어 구현에 적합한 고속 스트림 암호 AA32,” 한국통신학회 논문지, 제35권, 제6호, 2010. 6.

[7] 박창수, 조경연, “갈로이 선형 궤환 레지스터의 일반화,” 전자공학회논문지, 제43권, C1편, 제1호, 2006. 1.

[8] D. J. Bernstein, Synchronous Stream Cipher Salsa20, http://www.ecrypt.eu.org/stream/salsa20p3.html.

[9] “New European Schemes for Signatures. Integrity. and Encryption(NESSIE),” https://www.cosic.esat.kuleuven.be/nessie/.

[10] http://www.ecrypt.eu.org/stream/sw.html.

[11] http://www.ecrypt.eu.org/stream/hw.html.

[12] D. J. Bernstein, "Which phase-3 eSTREAM ciphers provide the best software speeds?," eSTREAM report 013, 2008.

[13] P. Souradyuti and B. Preneel, "Analysis of Non-fortuitous RC4 key stream generator,"

Progress in Crytology-INDOCRYPT, 2003.
[14] "Wireless LAN medium access control(MAC) and physical layer(PHY) specifications," Technical Report, IEEE Standard 802.11b, 1999.
[15] C. D. Canniere and B. Preneel, "Trivium Specifications," http://www.ecrypt.eu.org/stream/e2-trivium.html.
[16] E. Zenner, "Why IV Setup for Stream Cipher is Difficult," Proceedings of Dagstuhl Seminar on Symmetric Cryptography, 2007.
[17] 김길호, "대칭단을 이용한 암호와 복호가 다른 블록 암호의 재설계," 박사학위논문, 부경대학교 컴퓨터공학과, 2010.
[18] P. Hawkes and G. Rose, "Guess-and-determine attacks on SNOW," In Selected Areas in Cryptography - SAC 2002, LNCS vol. 2595, pp. 37 - 46, 2002.
[19] P. Hawkes and G. Rose, "Correlation cryptanalysis of SSC2," Presented at the Rump Session of CRYPTO, 2000.
[20] N. Courtois, "Fast Algebraic Attack on Stream Ciphers with Linear Feedback," Advances in Cryptology-CRYPTO 2003, LNCS vol. 2729, pp. 176-194, 2003.
[21] J. Nechvatal, E. Barker, L. Bassham, W. Burr, M. Dworkin, J. Foti and E. Roback, "Report on the development of the Advanced Encryption Standard(AES)," NIST report 106, 2001.
[22] J. Daemen, L. R. Knudsen and V. Rijmen, "The block cipher Square," Software Encryption, LNCS vol. 1267, pp. 149-165, 1997.

김 길 호 (Gil-Ho Kim)

- 정회원
- 한국방송통신대학교 전자계산학과 이학사
- 부경대학교 컴퓨터공학과 공학석사
- 부경대학교 컴퓨터공학과 공학박사
- 부경대학교 공과대학 IT융합응용공학과 시간강사
- 관심분야 : 암호 알고리즘, 컴퓨터 구조, 반도체회로 설계