블록 암호 알고리즘기반 의사난수발생기 제안과 안전성 분석

송 정 환[†]·현 진 수[††]·구 본 욱[†††]·장 구 영[††††]

요 약

신뢰성 있고 효율적인 정보보호를 위한 정보보호 시스템은 안전성이 입증된 암호기술을 사용하여 기밀성과 인증성을 보장 받아야 한다. 기본적으로 정보보호시스템이 갖추어야 할 요소인 난수 발생기는 수학적인 논리에 의하여 안전성과 난수성이 증명 가능하도록 설계되는 것이고, 암호알고리즘의 초기 변수값, 키 등을 생성하는데 이용되며 난수 발생기의 안전성이 전체 정보보호시스템의 안전성에 영향을 준다. 본 논문에서는 블록 암호 알고리즘을 기반으로 하는 의사난수 발생기를 설계하여 제안하고, 그의 안전성에 대해서 연구한다.

Proposing a PRNG based on a block cipher and cryptanalyzing its security

Junghwan Song[†]·Jinsu Hyun[††]·Bonwook Koo[†††]·Ku-Young Chang[††††]

ABSTRACT

Cryptographic applications, such as data confidentiality and authentication, must be used for secure data communications. PRNG (Pseudo-Random Number Generator) is a basic cryptographic component which is supposed to be satisfied by criteria that are provable security and randomness properties. PRNG is used for generating an initial value or key value of cipher and security of whole cryptographic module depends on the security of PRNG. In this paper, we introduce an PRNG based on a block cipher and prove their security.

키워드 : 난수 발생기(RNG), 의사난수 발생기(PRNG), 난수(Random Number), 안정성(Security)

1. 서 론

난수(Random number)는 그것이 생성되기 전에 예측할 수 없는 수이다. 만일 어떤 수가 $0, \cdots, 2^n-1$사이의 난수라고 가정하면, 2^{-n}의 확률보다 높은 확률로 난수를 예측할 수 없다. 또한, m개의 난수가 생성되어있고, $m-1$개의 난수에 대하여 알고 있다고 하더라도, m번째 수도 역시 2^{-n}의 확률보다 높은 확률로 m번째 수를 예측할 수 없어야 한다. 그러나, 이러한 참 난수(True random number)를 발생하는 것은 현실적으로 불가능해서, 난수와 구별하기 어려운 의사난수(Pseudo random number)를 생성하여 난수로 사용한다. 의사난수란 난수와 구별할 수 있는 효율적인 알고리즘이 존재하지 않는 수를 의미한다[1]. 이러한 의사난수를 생성하는 방법으로는 크게 하드웨어에서 얻을 수 있는 높은 엔트로피의 잡음원(반도체의 양자효과, 방사선 소스, 디스크 드라이브 내의 난기류 등등)을 간단한 필터링 후 이용하는 방법과 위의 잡음보다는 엔트로피가 낮은 잡음원(키 스트로크, CPU time 등등)에 해쉬 함수나 블록 암호 알고리즘을 적용하는 방법이 있다[2-6]. 전자의 경우는 생성된 의사난수가 참 난수와 같이 엔트로피가 매우 높다는 장점이 있는 반면에, 높은 엔트로피의 잡음을 얻기가 쉽지 않고, 다양한 플랫폼에 적용하기 힘들다는 단점이 있다[7,8]. 후자의 경우는 전자의 경우보다는 엔트로피가 낮다는 단점이 있으나, 키나 seed가 공개되지 않고, 해쉬 함수나 블록 암호 알고리즘이 분석되기 어렵다면 충분한 안전성을 보장할 수 있다. 또한, 여러 플랫폼에 적용가능하며, 원하는 길이의 의사난수 수열을 생성할 수 있다는 장점이 있다.

128비트 난수가 사용되고, 서버 및 PC와 같은 플랫폼에서 사용되는 인증알고리즘에서 요구되는 의사난수생성기는 하드웨어로 의사난수를 생성하기보다는 소프트웨어로 구현되는 것이 더 적합하다[9-11].

본 논문에서는 블록 암호 알고리즘을 기반으로 하는 랜덤함수를 사용한 의사난수 발생기를 제안하고, 의사난수 함수 조건 적합성, 블록암호 알고리즘의 암호분석적인 공격에 대한 저항성, 기존의 의사난수 발생기에 대한 공격에 대한 저항성, 난수성 검정 등을 통하여 그의 안전성에 대하여 분석한다.

※ 본 연구는 한국전자통신연구원(ETRI) 연구과제(01MK1110) 지원으로 수행하였습니다
† 정 회 원 : 한양대학교 수학과 교수
†† 정 회 원 : 한국정보보호진흥원 연구원
††† 준 회 원 : 한양대학교 대학원 수학과
†††† 정 회 원 : 한국전자통신연구원 선임연구원
논문접수 : 2002년 7월 24일, 심사완료 : 2002년 12월 20일

2. 의사난수 발생기 설계시 고려사항

소프트웨어 의사난수 발생기를 설계에 기본적으로 고려할 사항은 다음과 같다[6, 7].

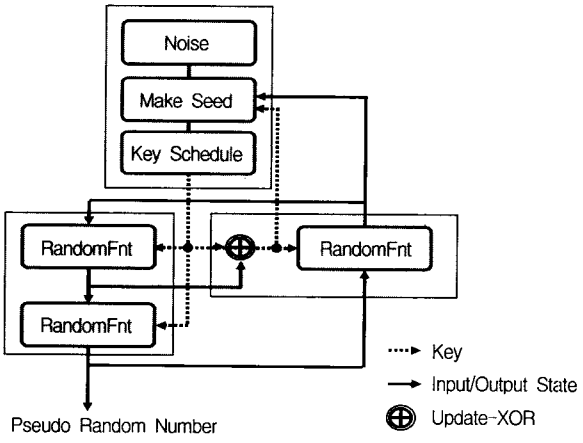
- seed를 추정하기 힘들도록 엔트로피를 높여야 한다.
- 키와 seed에 대한 관리를 해야한다.
- 의사난수 발생기와 그 과정에 대한 암호분석 공격에 저항성이 있어야 한다.
- 키가 노출 된 뒤에도, 역추적 공격에 저항성이 있어야 한다.
- 효율적이어야 한다.
- 긴 주기를 가져야 한다.

의사난수 발생기에 사용되는 키나 seed가 공개되면, 의사난수에서 생성되는 난수는 예측 가능하게 되어, 난수로서의 의미가 소멸된다. 그러므로, 기본적으로 의사난수 발생기가 의사난수를 생성하는 동안에는 그 키와 seed를 포함한 모든 내부변수들이 안전하게 저장된다는 가정이 있어야 한다. 그러나, 강력한 공격 모델을 설정하고자 할 경우는 새로운 난수를 생성하기 위해 메모리에 저장된 값 또는 그 일부분이 공격자에게 공개될 수 있다는 가정을 한다. 이러한 경우는 알고리즘의 구조적인 결함에 의한 것이 아니라, 구현 시 보완해야할 사항이다.

의사난수 발생기가 구현될 플랫폼이 PC와 같은 환경이 아닌 IC카드와 같은 하드웨어 환경일 경우도 고려하여야 하며, 이럴 경우는 이용할 수 있는 잡음원이 한정되어있다는 것을 고려하여서 설계하여야 한다[4].

3. 의사난수 발생기의 구조

본 논문에서 제안되는 의사난수 발생기는 크게 3개의 알고리즘으로 나누어진다. 첫 번째는 잡음 정보로부터 생성한 seed를 새롭게 갱신시키는 reseeding 알고리즘이고, 두 번째는 seed로부터 의사난수를 생성하는 랜덤화 알고리즘이며, 세 번째는 랜덤화 알고리즘의 입력 값을 새롭게 갱신시키는 업데이트 알고리즘이다.



(그림 1) 의사난수 발생기의 구조

초기에는 seed 값을 입력받아서 키를 생성하며, 이때 사용될 상수 C1(IV_{C1}), C2(IV_{C2})는 초기에 설정된 값을 이용한다. 그리고, 랜덤화 알고리즘에 사용될 state도 초기에 설정된 값(IV_{State})을 이용한다.

본 의사난수 발생기는 의사난수를 생성할 때마다 키를 다시 생성하는 것이 아니라, 기존에 생성된 키를 간단한 XOR 연산으로 업데이트 시킴으로서 효율성을 높이도록 설계되었다. 자세한 의사난수 발생기의 구조는 (그림 1)과 같다.

3.1 Reseeding 알고리즘

Seed의 엔트로피는 의사난수 발생기의 안전성의 매우 중요한 역할을 한다. 그러므로, seed의 엔트로피를 높이기 위해서 임의적으로 제어하거나 추측하기 힘든 잡음 정보를 seed를 생성하는데 이용함으로 seed의 엔트로피를 높여야 한다.

본 의사난수 발생기의 reseeding 알고리즘은 크게 잡음 정보로부터 seed를 생성하는 알고리즘과 seed를 이용해서 키를 생성하는 알고리즘으로 구성되어있다. 전자를 seed 생성 알고리즘, 후자를 키 생성 알고리즘이라고 하자.

3.1.1 Seed 생성 알고리즘(Make Seed)

Seed를 생성하기 위해서는 예측 또는 제어하기 힘든 잡음 정보를 이용한다. 프로그램이 구현되는 각 플랫폼에 따라 사용될 잡음을 수집하는 함수를 설정한다. 이때 수집된 잡음은 최소 128비트이어야 한다. 수집된 잡음을 균일하게 만들기 위해서 랜덤화 함수를 사용하는데, 이를 RandomFnt (state, key)라고 하자. state는 블록 암호 알고리즘의 평문에 해당하는 것으로 128비트이며, RandomFnt 함수의 출력도 128비트이다.

잡음 128비트와 마지막으로 업데이트된 state 128비트를 다음 (1)과 같이 64비트 I1, I2, I3, I4로 표시하자.

$$(\text{Noise} \parallel \text{state}) = (I1 \parallel I2 \parallel I3 \parallel I4) \quad (1)$$

$$(O1 \parallel O2) = \text{RandomFnt}(I1 \parallel I2, \text{key}) \quad (2)$$

$$(O3 \parallel O4) = \text{RandomFnt}(I2 \oplus O2 \parallel I3, \text{key}) \quad (3)$$

$$(O5 \parallel O6) = \text{RandomFnt}(I3 \oplus O4 \parallel I4, \text{key}) \quad (4)$$

각각 출력된 O1~O6 중에서 3개의 64비트 O1, O3, O5만을 출력한다. 64비트 O1은 키 생성 알고리즘에 사용되는 두 개의 32비트 상수 C1, C2로, (O3 || O5)는 새로운 128비트 seed로 사용한다.

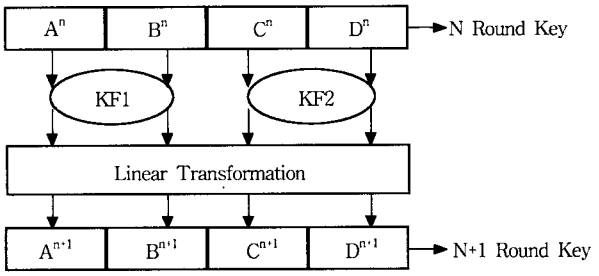
3.1.2 키 생성 알고리즘(Key Schedule)

키 생성 알고리즘은 32비트 워드 단위로 연산되며, 64비트를 입력으로 하는 KF함수와 선형변환으로 구성되어있다. 알고리즘의 구조는 (그림 2)와 같다.

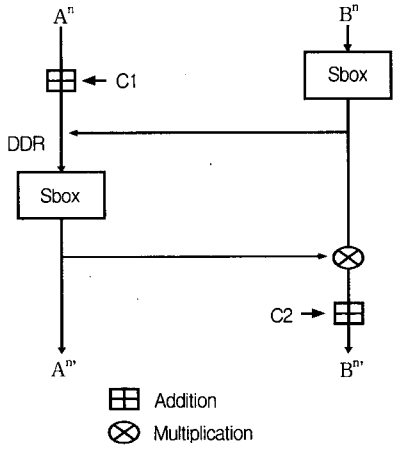
KF함수의 내부는 2개의 table lookup S-box와 32비트 상에서의 2번의 덧셈과 1번의 DDR(Data Dependant Rotation)과 1번의 곱셈으로 구성되어 있으며, 이를 (그림 3)에 나타

내었다. S-box는 RandomFnt에 사용되는 S-box를 이용하며, 덧셈과 곱셈의 법은 2^{32}이다. DDR연산은 B^n을 S-box로 계산한 4바이트 결과의 MSB(Most Significant Bit) 4개로 4비트를 구성하여 왼쪽으로 회전시킨다. 키 값의 일부가 알려졌을 경우에 seed를 알아내기 힘들도록 하기 위하여 전단사 함수가 아닌 곱셈 연산이 사용된다. KF함수에 사용되는 상수 C1, C2는 seed 생성 알고리즘으로 생성한다.

KF함수의 출력인 4개의 워드는 다시 4×4 선형변환을 거쳐서 각 라운드 키를 출력하게 된다. 사용된 선형변환은 최대 branch number 4를 갖는 4×4 선형변환을 사용한다.



(그림 2) 키 생성 알고리즘의 구조



(그림 3) KF함수의 구조

3.2 랜덤화 알고리즘

3.2.1 랜덤화 알고리즘의 구조

의사난수 발생기에 대한 여러 공격에 안전하게 하기 위하여, 아래의 RandomFnt 함수를 두 번 반복하여서 의사난수를 생성하도록 한다. 알고리즘을 식으로 나타내면 다음 식 (5), 식 (6)과 같다.

$$\text{Rand 1} = \text{RandomFnt(state, key)} \quad (5)$$

$$\text{Rand 2} = \text{RandomFnt(Rand1, key)} \quad (6)$$

Rand2가 출력되는 의사난수이고, Rand1은 업데이트 알고리즘의 입력으로 사용된다.

3.2.2 RandonFnt 의 구조

랜덤화 알고리즘의 RandomFnt는 3-Feistel 구조로 되어 있다. 내부의 F함수는 SPN 구조로 되어있어서, DC/LC등의 공격에 대한 안전성을 계산할 수 있다.

state는 블록 암호 알고리즘의 평문에 해당하는 값으로, 키와 더불어 RandomFnt 함수의 입력으로 사용된다. state의 크기는 128비트이며, 키의 크기는 128×round 비트이다. 기본적으로 랜덤화 알고리즘은 8라운드로 되어있으므로, 키의 크기는 128×8 = 1024비트가 된다. 그러나, 키가 128비트 seed로부터 생성되기 때문에 RandomFnt 함수의 전수조사에 대한 안전성은 seed의 크기에 의존하게 된다.

state를 각각 32비트 워드 (A_0, B_0, C_0, D_0)로 분할하고, r 라운드에서의 중간 값을 (A_r, B_r, C_r, D_r)으로 표시하자. 또한 각각의 32비트 워드에 대응하는 서브키를 (K^A_{r-1}, K^B_{r-1}, K^C_{r-1}, K^D_{r-1})와 같이 표시하자.

r = 1, ⋯ , 8일 때, RandomFnt 함수를 식으로 나타내면 다음 식 (7)~식 (10)과 같다.

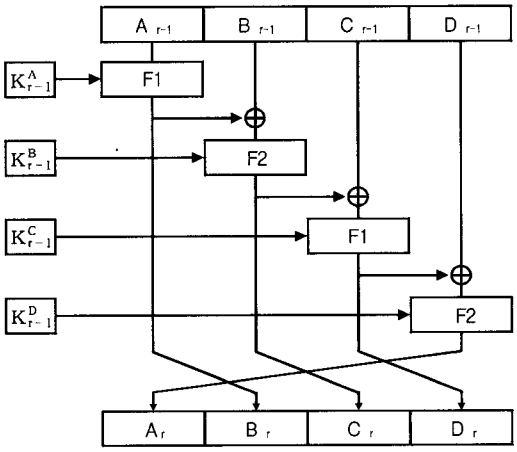
$$B_r = F(K^A_{r-1}, A_{r-1}) \quad (7)$$

$$C_r = F(K^B_{r-1}, B_{r-1} \oplus B_r) \quad (8)$$

$$D_r = F(K^C_{r-1}, C_{r-1} \oplus C_r) \quad (9)$$

$$A_r = F(K^D_{r-1}, D_{r-1} \oplus D_r) \quad (10)$$

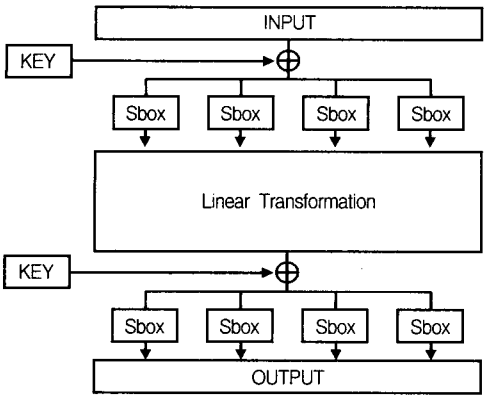
위의 식을 그림으로 나타내면 다음 (그림 4)와 같다.



(그림 4) RandomFnt 함수의 라운드 구조

3.2.3 F 함 수

F함수의 구조는 2라운드 SPN 구조이다. 2라운드 SPN 구조는 두 번의 키 XOR과 두 번의 table lookup S-box 와 1번의 선형변환으로 구성되며, 차분 공격이나 선형 공격과 같은 기존의 알려진 공격에 대한 안전성이 증명 가능하다. 홀수 번째 워드와 짝수 번째 워드에 사용되는 F함수를 각각 F1, F2라고 하며, 각각 다른 S-box와 선형변환을 사용한다. 그러나, 하나의 F함수의 내부에 두 번씩 사용되는 키와 S-box는 동일한 것을 반복 사용한다. F함수의 구조를 살펴보면 다음 (그림 5)와 같다.



(그림 5) F함수의 구조

가). S-box

1). S-box 설계 원칙

의사난수 발생기가 내부적으로 블록 암호 알고리즘의 구조로 되어있으므로, S-box의 취약성을 이용하여서 의사난수 발생기를 공격하게 된다. 이에 기존의 알려진 취약성을 보완하여 S-box를 안전하게 설계하고자 한다.

S-box의 설계 원칙은 다음과 같다.

① 차분 공격과 선형 공격에 안전하다.
② 대수적 차수가 높다.
③ 고정점이 없다.
④ 전단사 함수이다.

$GF(2^8)$상에서 정의된 함수를 고려해보면, $\gcd(k, 2^8-1) = 1$인 k를 선택하면 아래 함수는 전단사 함수가 된다.

$$GF(2^8) \rightarrow GF(2^8) \ : \ x \mapsto x^k$$

또한, 짝수 차원 8에서 정의된 S-box의 최대 차분 확률과 최대 선형 확률은 각각 $\frac{1}{2^6}$이다. 위의 확률을 갖는 S-box를 선택하여야 하고, 고계 차분 공격에 안전하기 위해서는 S-box의 성분함수의 대수적 차수가 커야 한다.

위의 조건을 만족하는 함수 중 $x^{-1} = x^{254}$이 있다. 그러나, 이 함수는 0을 0으로 1을 1로 대응시키는 고정점이 발생한다. 이 문제를 해결하기 위해 $\{0,1\}^8$상의 affine 변환을 작용한다. Affine 변환은 전단사, DC, LC확률 그리고 차수의 성질을 변화시키지 않는다.

2). S-box 설계 방법[12]

의사 난수 발생기에 사용되는 두 S-box S1, S2의 설계 방법은 다음과 같다.

① $m(x) = x^8 + x^6 + x^5 + x^4 + 1$($171_x$에 대응)을 이용하여 $GF(2^8)$ 설계
② 갈로아 체 $GF(2^8)$ 상에서 변환 $x^{-1} = x^{254}$ 선택
③ 다시 $\{0,1\}^8$ 상에서 affine 변환 작용

고정점을 제거하기 위해서 affine 변환 식 (11)을 적용시킨다.

A가 8×8 정칙행렬일 때,

$$y = Ax \oplus b \tag{11}$$

S1을 생성할 때 사용한 행렬 A와 상수 b는 다음 식 (12)와 같다.

$$\begin{pmatrix} y_7 \\ y_6 \\ y_5 \\ y_4 \\ y_3 \\ y_2 \\ y_1 \\ y_0 \end{pmatrix} = \begin{pmatrix} 0&0&1&1&0&0&1&1 \\ 0&1&1&1&1&0&0&1 \\ 0&1&1&0&1&0&1&1 \\ 0&0&1&0&0&1&1&1 \\ 1&1&1&0&0&0&1&1 \\ 1&0&1&0&1&0&1&1 \\ 0&0&0&1&1&1&1&0 \\ 0&1&0&0&1&1&1&0 \end{pmatrix} \begin{pmatrix} x_7 \\ x_6 \\ x_5 \\ x_4 \\ x_3 \\ x_2 \\ x_1 \\ x_0 \end{pmatrix} \oplus \begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \tag{12}$$

S2을 생성할 때 사용한 행렬 A와 상수 b는 다음 식 (13)과 같다.

$$\begin{pmatrix} y_7 \\ y_6 \\ y_5 \\ y_4 \\ y_3 \\ y_2 \\ y_1 \\ y_0 \end{pmatrix} = \begin{pmatrix} 0&1&0&1&0&1&1&1 \\ 0&1&1&1&0&0&1&1 \\ 1&1&1&0&1&0&0&0 \\ 0&0&0&1&1&1&0&0 \\ 1&0&1&0&0&1&0&0 \\ 1&0&0&1&1&1&1&1 \\ 0&0&0&1&0&1&1&0 \\ 0&0&1&0&0&1&0&1 \end{pmatrix} \begin{pmatrix} x_7 \\ x_6 \\ x_5 \\ x_4 \\ x_3 \\ x_2 \\ x_1 \\ x_0 \end{pmatrix} \oplus \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} \tag{13}$$

나). 선형변환

4×4 선형변환으로 active S-box의 개수의 최소값이 최대인 4가 되는 선형변환을 각각 F1, F2함수의 선형변환 L1, L2로 선택한다[13].

각각의 선형변환은 다음 식 (14)와 같다.

$$L1 = \begin{pmatrix} 1&0&1&1 \\ 0&1&1&1 \\ 1&1&0&1 \\ 1&1&1&0 \end{pmatrix}, \quad L2 = \begin{pmatrix} 1&1&1&0 \\ 1&1&0&1 \\ 0&1&1&1 \\ 1&0&1&1 \end{pmatrix} \tag{14}$$

3.3 업데이트 알고리즘

업데이트 알고리즘에서 랜덤화 알고리즘의 출력 값을 이용하므로, 랜덤화 알고리즘의 구조를 다시 보면 다음 식 (15), 식 (16)과 같다.

$$\text{Rand 1} = \text{RandomFnt(state, key)} \tag{15}$$
$$\text{Rand 2} = \text{RandomFnt(Rand1, key)} \tag{16}$$

Rand2는 생성된 의사난수로 사용될 값이나 공격자에게 쉽게 노출될 수 있다. 만일 키도 새롭게 업데이트 되지 않고 공격자에게 노출된다면, RandomFnt 함수가 역으로 계산될 수 있게 된다. 그러므로, 난수를 생성한 후 키와 state를 제외한 모든 내부변수들은 모두 0으로 초기화 시켜야하며, 키는 새롭게 업데이트 하여야 한다.

기호 ⊕로 표시된 Update-XOR 연산은 키를 각 라운드에 사용되는 128비트씩 128비트 Rand1과 XOR하는 것이다. 이러한 과정으로 키는 업데이트 되며, 업데이트 된 키와 Rand 2를 RandomFnt 함수의 입력으로 한 결과로 state를 업데이

트 시킨다. 식으로 나타내면 다음 식 (17), 식 (18)과 같다.

$$key = key \oplus output1 \quad (17)$$

$$state = RandomFnt(\ Rand2,\ key\ ) \quad (18)$$

4. 의사난수 발생기의 안전성[7, 14-16]

의사난수 발생기의 안전성은 다음과 같은 조건을 고려한다.

- 의사난수 함수 조건 적합성
- 블록 암호 알고리즘의 암호 분석적인 공격(DC/LC)에 대한 저항성
- 기존의 의사난수 발생기에 대한 공격에 대한 저항성
- 난수성 검정

4.1 의사난수 발생기에 대한 안전성

4.1.1 Seed에 대한 전수조사

Reseeding 과정에서 사용되는 seed가 128비트이며, 알고리즘 내부적으로 사용되는 변수들도 128비트이므로, 전수조사에 안전하다.

4.1.2 선택/기지 입력공격

만일 공격자가 seed를 생성하기 위해 필요한 잡음 정보에 대하여 특정 값으로 조작하거나, 예측할 수 있다면 seed에 대한 전수조사의 복잡도가 2^{128}보다 작게된다. 그러므로, 잡음 정보를 선택할 때, 공격자가 조작, 예측하기 힘든 잡음 정보를 이용하여야 한다.

또한, seed를 생성하는 알고리즘에 RandomFnt 함수를 이용하기 때문에, 완벽하게 잡음원에 대한 정보를 모르는 경우, 입력의 아주 작은 차이로도 생성되는 seed에 대한 정보를 알기 어렵다.

4.1.3 기지 출력 공격

일반적으로 공격자는 의사난수 발생기에 의해 출력되는 의사난수를 안다고 가정하자. 그러나, 의사난수 발생기는 RandomFnt 함수를 이용하여 의사난수를 발생하므로, 공개된 출력으로 입력 state 또는 키를 알아내기 위해서는 RandomFnt 함수를 분석해야만 한다. 또한, 의사난수가 발생한 뒤, 키와 state를 업데이트 알고리즘을 이용하여서 업데이트 시키고, 나머지 내부변수들은 모두 초기화시키므로, 공개된 출력 값만으로 다음 랜덤화 알고리즘의 입력을 알 수 없다. 이를 알기 위해서도 RandomFnt 함수를 분석하여야만 한다. 즉, RandomFnt 함수를 분석할 수 없으면, 공개된 출력만으로 기존의 입력이나 다음의 입력 값에 대한 어떠한 정보도 얻거나 추측할 수 없다.

4.1.4 추적/역추적 공격[16]

의사난수 발생기 프로그램상의 허점을 이용한 공격이 존재하여, 공격자가 의사난수 발생기의 내부변수를 알아낼 수 있으나, 의사난수 발생기가 동작하는 동안의 내부변수는 접근할 수 없으므로, 난수를 생성한 후의 다음 난수를 생성하기 위한 내부변수에 대하여는 접근할 수 있다고 가정하자. 즉, 하나의 의사난수를 생성한 후 다음의 의사난수를 생성하기 위해 필요한 키와 state는 안전하지 않다는 가정이다.

만일, 위와 같이 키와 state가 공격자에게 노출되었을 경우, 그 후에 생성되는 의사난수는 모두 공개가 될 수밖에 없다. 그래서, 의사난수 발생기가 일정크기의 의사난수를 생성하거나, 일정 시간이 지난 뒤, 자동으로 reseeding 알고리즘이 동작하도록 하여 공개되는 의사난수를 최소한으로 줄여야한다. 또한, RandomFnt 함수를 역으로 계산하여 seed를 계산하거나, 과거에 생성된 의사난수가 공개되는 것을 막기 위해서, 랜덤화 알고리즘에 RandomFnt 함수를 두 번 사용하고, 첫 번째 RandomFnt 함수의 결과를 업데이트 알고리즘에 사용한다.

4.2 RandomFnt 함수의 안전성

4.2.1 의사난수 함수

동일한 키일 때, 입력(state)이 다른 경우, 각각 생성된 의사난수를 구별할 수 있는 확률은 2^{-128}이어야만 한다. RandomFnt 함수는 2라운드 후 이 조건을 만족시킨다.

먼저, 키는 동일하게 적용되기 때문에, 생략하도록 하고, F함수는 하나만 사용하고, F함수의 결과는 난수라고 가정하자. 즉, F함수의 입력이 다른 경우, 각각을 구별할 수 있는 확률은 2^{-32}이다.

state가 다른 경우는, 각각의 32비트 워드 A_0, B_0, C_0, D_0 중 최소한 한곳이 다르다는 것을 의미한다. 그러므로, 2라운드 후 결과가 A_0, B_0, C_0, D_0 모두에 영향을 받고, 그 값들이 F함수의 입력이 될 때, F함수의 결과인 A_2, B_2, C_2, D_2를 구별할 수 있는 확률은 각각 2^{32}이다.

$$A_2 = F(D_2 \oplus D_1) \quad (19)$$

$$\begin{aligned} D_1 &= F(C_0 \oplus C_1) = F(C_0 \oplus F(B_0 \oplus B_1)) \\ &= F(C_0 \oplus F(B_0 \oplus F(A_0))) \end{aligned} \quad (20)$$

$$\begin{aligned} D_2 &= F(C_1 \oplus F(B_1 \oplus F(F(D_1 \oplus D_0)))) \\ \text{즉, } A_2 &= F(D_2 \oplus D_1) \\ &= F[F(C_1 \oplus F(B_1 \oplus F(F(D_1 \oplus D_0)))) \\ &\quad \oplus F(C_0 \oplus F(B_0 \oplus F(A_0)))] \end{aligned} \quad (21)$$

만일 다른 state 값 $P_i = (A_{i0}, B_{i0}, C_{i0}, D_{i0})$, $P_j = (A_{j0}, B_{j0}, C_{j0}, D_{j0})$로부터 얻은 결과 중 첫 번째 32비트 워드를 A_{i2}, A_{j2}라고 할 때,

$$Pr(A_{i2} = A_{j2} \mid P_i \neq P_j) = 2^{-32} \quad (22)$$

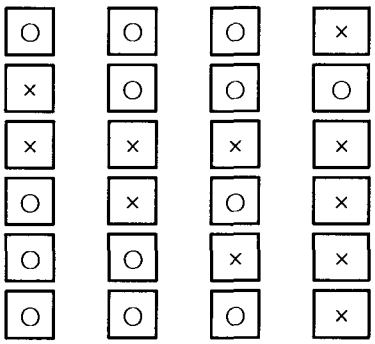
그러므로, 비슷한 방법으로 B_2, C_2, D_2의 경우에도 A_2와 같이 표시할 수 있고, 이는 각각의 32비트 워드 A_2, B_2, C_2, D_2를 구별하기 할 수 있는 확률은 2^{32}라는 것을 의미한다. 즉, RandomFnt 함수는 2라운드에서 의사난수 함수의 성질을 만족한다.

4.2.2 Differential Cryptanalysis[17]

RandomFnt 함수의 내부에 사용되는 F함수 F1과 F2는 최소한 Active S-box를 4개 이상 포함하도록 설계되어 있다. 그리고, F함수에 사용된 S-box는 최대 차분 확률이 최대 2^{-6}이 되도록 선택되었다. 그러므로, F함수의 DC에 대한 확률은 최대 2^{-24}를 넘지 못한다.

각 라운드에서 최소한 1개의 F함수에 입력 차분이 0이 아니라고 가정하면, 6라운드면 2^{-144}의 안전성을 이론적으로 보장받게된다. 그러나, 각 라운드에서 하나의 active F함수가 연속하여서 생기는 경우는 발생하지 않는다. 그리고, DC확률의 상한값을 계산하여 보면, 다음과 같은 5라운드 반복 차분 특성을 찾을 수 있다. 이때 10개의 F함수에 입력 차분이 0이 아니므로, 5라운드 반복 차분 확률은 2^{-240}이 된다. 그러므로, 5라운드 이상이면 RandomFnt 함수가 DC에 충분히 안전하다.

다음 (그림 6)은 입력 마지막 워드에 차분이 발생하였을 경우, 가능한 5라운드 반복 차분 구조이다. ×로 표시된 워드는 차분이 0이 아닌 것이고, ○로 표시된 워드는 차분이 0인 것이다.



(그림 6) 5라운드 차분 구조

4.2.3 Linear Cryptanalysis[18]

RandomFnt함수의 내부에 사용되는 F함수 F1과 F2는 최소한 Active S-box를 4개 이상 포함하도록 설계되어있다. 그리고, F함수에 사용된 S-box는 최대 선형근사 확률이 최대 2^{-4}가 되도록 선택되었다. 그러므로, F함수의 LC에 대한 확률은 최대 2^{-16}을 넘지 못한다.

DC와 유사하게 5라운드 선형 근사식을 찾을 수 있고 10개 F함수가 active할 경우, LC에 대한 확률은 최대 2^{-140}을 넘지 못하게 된다. 그러므로, RandomFnt 함수는 LC에 대하여도 충분히 안전하다.

4.3 난수성 검정[19]

4.3.1 난수성 검정 종류

(1) Frequency test
(2) Frequency test within a block
(3) Runs test
(4) Test for the longest run of ones in a block
(5) Binary matrix rank test
(6) Discrete fourier transform(Spectral) test
(7) Non-overlapping template matching test(길이가 9인 148개의 서로 다른 Template으로 분류)
(8) Overlapping template matching test
(9) Mauer's test
(10) Lempel-Ziv Compression test
(11) Linear complexity test
(12) Serial test(Subblock의 길이에 따라 2가지로 분류)
(13) Approximate entropy test
(14) Cumulative sums test(덧셈의 방향에 따라 Forward, Backward로 분류)
(15) Random excursions test(State에 따라 8가지로 분류)
(16) Random excursions variant test(State에 따라 16가지로 분류)

4.3.2 난수성 검정 결과

난수 테스트를 적용하기 위한 데이터는 주기적으로 Re-seeding 알고리즘이 적용되는 것과 그렇지 않는 것을 사용하여 생성하였으며, 각각의 데이터를 생성하기 위한 초기 입력으로는 0과 난수를 사용하였다. Reseeding 알고리즘은 난수 1000개를 생성할 때마다 알고리즘이 적용되도록 하였다.

각각의 표본은 300개의 1,048,576(= 2^{20})비트 수열로 구성되어 있다. 유의수준 α는 0.01을 사용하였다.

- Data 1 : 초기 입력 0, Reseeding 알고리즘 사용 안함.
- Data 2 : 초기 입력 난수, Reseeding 알고리즘 사용 안함.
- Data 3 : 초기 입력 0, Reseeding 알고리즘 사용함.
- Data 4 : 초기 입력 난수, Reseeding 알고리즘 사용함.

가). 기각 수열 개수 분석

다음 <표 1>은 4가지 종류의 300개의 표본들에 대하여 검정을 실시하였을 때, 각 검정별로 300개의 표본들 중 기각된 표본의 개수를 나타낸 것이다. 각 검정별로 300개의 표본 중 기각되는 표본의 개수가 11개 이하이면 난수성을 만족한다[19, 20]. 최대로 기각되는 수열의 개수가 Data 2의 Excursion Variant Test에서 State가 -8인 경우 10임을 알 수 있다. 따라서, 최대 허용 기각수 11개보다 적다.

Nonoverlapping Template 검정의 경우 검정의 특성상 148가지의 서로 다른 Template의 형태에 대하여 검정을 실시한 것이며, Serial 검정의 경우 Subblock 의 길이에 따라 1-Serial 검정과, 2-Serial 검정으로 나뉘고, Cusum 검정의 경우는 덧셈의 방향에 따라 Forward와 Backward로 나뉘며, Excursion 검정은 State에 따라서, -4에서 4까지 0을 제외한 8가지 경우, Excursion Variant 검정은 -8에서 8까지 0을 제외한 16가지 경우로 나뉘어 진다. <표 1>과 <표 2>의 괄호안의 표시는 나뉘어진 검정들 중 최대 기각수를 가지는 경우를 표시한 것이다.

나). p-value 들의 분포 분석

AES Round 2와 같이 표본들의 p-value 분포가 [0, 1] 사

〈표 1〉 검정별 최대 기각표본 개수

Test	Data 1	Data 2	Data 3	Data 4	Test	Data 1	Data 2	Data 3	Data 4
Frequency	3	5	0	1	Universal	0	9	1	5
Frequence M	4	3	4	4	Lempel-Ziv	4	3	3	5
Run	1	4	3	1	Linear Complexity	2	0	3	2
Longest Run	2	6	3	1	Serial	3 (1)	6 (1)	3 (1)	4 (1)
Rank	9	0	1	6	Approximate Entropy	4	2	5	4
DFT	1	0	1	1	Cusum	3 (f)	5 (f)	1 (f)	3 (f)
Nonoverlapping Template	8 (110101000)	8 (010001111)	8 (110101000)	9 (110111100)	Excursion	6 (-4)	6 (-4)	9 (4)	7 (-4)
Overlapping	7	2	3	2	Excursion Variant	9 (-8)	10 (-8)	9 (-5)	6 (7)

〈표 2〉 검정별 최소 p-value 의 p-value

Test	Data 1	Data 2	Data 3	Data 4	Test	Data 1	Data 2	Data 3	Data 4
Frequency	0.946307682	0.159612843	0.062820705	0.637119423	Universal	0.275708934	0.845772902	0.110952146	0.574903493
Frequence M	0.474985665	0.991467609	0.110952177	0.602457598	Lempel-Ziv	0.060238695	0.000817567	0.032922834	0.061517407
Run	0.245071627	0.520767416	0.685578614	0.69245503	Linear Complexity	0.581769534	0.952777715	0.949602001	0.319083548
Longest Run	0.602457648	0.834308264	0.494391641	0.822533725	Serial	0.588651795 (2)	0.383826586 (2)	0.240913618 (2)	0.474985665 (1)
Rank	0.540878431	0.175048507	0.319083476	0.86769168	Approximate Entropy	0.202267695	0.304125812	0.33453822	0.878107395
DFT	0.91996686	0.165646132	0.851382602	0.205896529	Cusum	0.117660727 (f)	0.602457598 (b)	0.449672005 (b)	0.588651844 (f)
Nonoverlapping Template	0.000474449 (000001111)	0.000540415 (111110110)	0.000631473 (111101110)	0.008266274 (000111111)	Excursion	0.139920675 (-3)	0.275708934(-2)	0.28495917 (1)	0.102525697 (3)
Overlapping	0.035173539	0.547636976	0.867691718	0.71974656	Excursion Variant	0.021998606 (-6)	0.026056713(-8)	0.049770276 (-4)	0.138596186 (-5)

이에서 균일하게 분포되어 있는가를 검정하기 위해서 자유도 9인 χ^2 분포를 바탕으로 p-value 들의 p-value를 다음과 같이 계산한다.

$$\chi^2 = \sum_{i=0}^{10} \frac{F_i - \frac{300}{10}}{\frac{300}{10}} \qquad (23)$$

F_i : # of p-values in subinterval i

다음 <표 2>는 4가지 종류의 300개 표본들에 대하여 검정을 실시하였을 때의 p-value 들의 p-value를 식 (23)에 의해 계산한 것을 나타낸 것이다.

평가기준[19]와 같이 계산된 p-value가 0.0001보다 작을때 난수성을 만족하지 않는다고 한다면, Data1의 Nonoverlapping Template Test에서 전체 검정 테스트 중 가장 작은 값 0.000474449가 나왔으나, 기준 0.0001보다 크다.

5. 결 론

신뢰성 있고 효율적인 정보보호 시스템이 폭 넓게 사용되기 위해서는 안전성이 보장된 암호기술 적용이 필수적이다. 이러한 정보보호 시스템에서 기밀성 기능과 인증성 기능이 실현되기 위하여 암호 알고리즘의 입력 키 생성과 시스템 변수 생성 혹은 Random challenge 등에 활용할 수 있는 난수 발생기가 적용되어야 한다. 따라서 난수 발생기는 정보보호 시스템이 갖추어야 할 기본적인 요소이며, 수학적인 논리에 의하여 안전성이 증명 가능하도록 설계되어야 하며, 광범위하게는 암호 알고리즘이라고 할 수 있다.

본 논문에서 인증 알고리즘에 적합하도록 구현환경과 입출력 방식 등을 고려하여 소프트웨어로 구현이 용이하도록 의사난수 생성 알고리즘을 제안하였다.

제안된 의사난수 발생기의 내부 함수 RandomFnt는 차분공격, 선형 근사 공격과 같은 암호 분석에 안전하며, 전체적인 구조는 선택 평문 공격, 기지 출력 공격 등과 같이 기존에 알려진 공격들에 안전하도록 설계되었다. 또한, 생성된 의사난수 수열의 난수성을 알아보기 위해 189개의 통계적 난수 검정을 시행하여 미국에서 AES 선정기준중 하나인 난수성 평가기준으로 제시한 모든 기준을 통과하는 결과를 얻었다.

외국 기술에 의존하지 않고 국내 독자적인 기술력으로 안

전성이 입증되도록 개발한 본 난수 발생기는 민간용 정보보호 시스템에 적용되어 활용될 것이라 기대한다[21].

참 고 문 헌

[1] P. Gutmann, "Software Generation of Practically Strong Random Numbers, extended version," available at http://www.cryptoengines.com/~peter/06 random.pdf, June, 2000.

[2] D. Davis, R. Ihaka and P. Fenstermacher, "Cryptographic Randomness from Air Turbulience in Disk Drives," Advances in Cyrptology-CRYPTO'94 Proceedings, Springer-Verlag, pp.114-120, 1994.

[3] G. B. Agnew, "Random Source for Cryptographic Systems," Advances in Cryptology-EUROCRYPT '87 Proceedings, Springer-Verlag, pp.77-81, 1988.

[4] RSA Data Security, "Hardware-based random number generation," An RSA Data Security Wjite Paper, http://developer.intel.com/design/security/rng/rngppr_v3.htm.

[5] Intel Platform Security Division, The Intel random number generator, available at http://developer.intel.com/design/security/rng/rngppr_v3.htm, 1999.

[6] R. C. Fairchild, R. L. Mortenson and K. B. Koulthart, "An LSI Random Number Generator (RNG)," Advances in Cryptology-CRYPTO'84 Proceedings, Springer-Verlag, pp. 203-230, 1985.

[7] 임채훈, 황효선, 강명희, "소프트웨어 의사난수 발생기의 설계 및 구현", CISC, pp.362-374, 2000.

[8] NIST, FIPS PUB 180-1, Secure Hash Standard, http://www.itl.nist.gov/div897/pubs/fip180-1.htm, April, 1995.

[9] 3GPP TSG SA WG3 Security-S3#15, "Report on the Evaluation of 3GPP Standard Confidentiality and Integrity Algorhthms," Sep., 2000.

[10] 3GPP TS 35.201 ; F8 and F9 Algorithms Specification; this is available at http://www.etsi.org/dvbandca/3gpp/3gppspecs.htm.

[11] B. E.Jung, H. Ryu, K. Kim, K. Y. Chang and O. Y Yi, "Analysis and Implementation for 3GPP Authentication Mechanism," Proceeding of WISA, pp.87-102, 2001.

[12] 송정환, 성수학, 서창호, "안전성 증명 가능한 블록 암호 알고리즘 개발 (I)", 한국정보보호센터 연구 00-5, 2000.

[13] 성수학, 박상우, 강주성, 지성택, "SPN 구조에서 최적의 선형 변환을 찾는 알고리즘", WISC2000, pp.189-197, 2000.

[14] J. Kelsey, B. Schneier and N. Ferguson, "Yarrow-160 : Notes on the Design and Analysis of the Yarrow Cryptographic Pseudorandom Number Generator," Selected Areas in Cryptography, SAC '99, LNCS 1758, Springer-Verlag, pp.13-33, 1999.

[15] J. Kelsey, B. Schneier, D. Wagner and C. Hall, "Cryptanalytic Attacks on Pseudorandom Number Generators," Fast Software Encryption, 5th International Workshop Proceedings, Springer-Verlag, pp.168-188, 1998.

[16] P. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and other Systems," Advances in Cryptology-CRYPTO'96 Proceedings, Springer-Verlag, pp.104-113, 1996.

[17] E. Biham, A. Shamir, "Differential Cryptanalysis of DES-like Cryptosystem," Advance in Cryptology-CRYPTO'90, LNCS, Springer-Verlag, Vol.537, pp.2-21, 1990.

[18] M. Matsui, "Linear Cryptanalysis method of DES cipher," Advance in Cryptology-EUROCRYPT'93, LNCS, Springer-Verlag, Vol.1039, pp.386-397, 1993.

[19] NIST Special Publication 800-22, "A Statistical Test Suite for the Validation of Random Number Generators and Pseudo Random Number Generators for Cryptographic Applications," July, 2000.

[20] 이상진, 성수학, 송정환, "블록 암호 알고리즘 구조 복잡도 및 암호문 특성분석", 한국정보보호센터 연구 00-8, 2000.

[21] 송정환, 조용국, 현진수, 신교일, "3GPP 인증알고리즘 적용위한 난수발생기 안전성 분석에 관한 연구", 한국전자통신연구원(ETRI) 연구 01MK1110, 2001.

송 정 환

e-mail : camp123@hanyang.ac.kr

1984년 한양대학교 수학과(학사)

1989년 Syracuse University Mathenatics (이학석사)

1993년 Rensselaer Polytechnic Institute Mathematics (이학박사)

1999년~현재 한양대학교 수학과 조교수

관심분야 : 암호학, 최적론, 수리계획법, 정보보호 이론

현 진 수

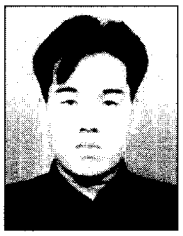
e-mail : jshyun@kisa.or.kr

2000년 한양대학교 수학과(학사)

2002년 한양대학교 수학과(이학석사)

2002년~현재 한국정보보호진흥원 연구원

관심분야 : 암호학, 정보보호 이론



구 본 욱

e-mail : kidkoo@ihanyang.ac.kr

2001년 한양대학교 수학과(학사)

2001년~현재 한양대학교 수학과 석사과정

관심분야 : 암호학, 정보보호 이론

장 구 영

e-mail : jang1090@etri.re.kr

1995년 고려대학교 수학과(학사)

1997년 고려대학교 수학과(이학석사)

2000년 고려대학교 수학과(이학박사)

2000년~현재 한국전자통신연구원 선임연구원

관심분야 : 정보보호 이론, 이동통신 보안, 대수학