情 報 保 護 學 會 誌
第 16 卷 第 4 號, 2006. 8

난수발생기의 현황 및 안전성 분석 기술 동향

강 주 성*

요 약

정보보호 시스템에 사용되고 있는 암호적 난수발생기의 현황을 조사하고 안전성 분석에 대한 최신 기술 동향을 살펴본다. 하드웨어를 기반으로 하는 난수발생기의 종류 및 사용 실태와 안전성 분석 기술을 조사 분석하며, 소프트웨어 기반 난수발생기에 대한 최신 설계 기술을 정리해 본다. 또한, 난수발생기의 출력 수열에 대한 통계적 안전성 평가 방법과 난수발생기 알고리즘에 대한 이론적 안전성 분석 기술에 대해서 논한다.

Ⅰ. 서 론

난수성(randomness)은 각종 암호 알고리즘을 설계하는 데에 있어서 가장 기본적이면서 중요한 항목이다. 그렇기 때문에 난수발생기는 거의 모든 암호 시스템에서 불가결한 기초 단위가 된다. 좋은 암호 시스템에서는 반드시 좋은 난수발생기를 요구하게 되어 있다.

대부분의 암호 프로토콜은 공격자에게 알려지지 않아야만 하는 비밀 값을 사용하기 때문에 이 값의 생성을 요구하게 되며, 이러한 값들의 생성은 난수발생기로부터 출발한다. RSA, DSA, Diffie-Hellman 등의 공개키 암호 시스템에서 필요한 소수의 생성을 위해서는 난수가 필요하다. DES, AES 등의 대칭키 암호 알고리즘에 사용되는 비밀키와 초기치의 생성을 위해서도 난수발생기가 사용되며, 운영모드에 사용되는 초기치 벡터의 값도 난수이기를 요구한다. 또한, 각종 식별(identification) 및 인증(authentication) 프로토콜에 사용되는 도전(challenge), 논스(nonce), 소금(salt) 등을 생성할 때에도 난수발생기가 사용된다.

난수발생기가 암호 시스템에서 필수적인 요소인 만큼 난수발생기에 조그마한 안전성적 결함이 있더라도 이는 대단한 파급효과를 지니게 된다. 그 대표적인 예로 미국 표준 디지털 서명 알고리즘인 DSA에 사용되었던 난수발생기의 결함[1]과 Netscape사의 난수발생기에 대한 취약성[2] 관련 보고를 들 수 있다.

논문으로 편찬되지는 않았지만 미국 벨연구소의 Bleichenbacher는 미연방 표준 FIPS 186-1에 나타나 있는 난수발생기의 비균일성(non-uniformity)을 발견하였다. 그의 공격은 2^{64}의 공격량과 2^{22}개의 기지 서명값들(known signatures)을 필요로 하는 것으로 DSA에 사용되는 난수발생기의 안전성에 문제가 있음을 지적한 것이다. DSA가 금융 거래 등에서 널리 사용되고 있기 때문에 이 사실은 언론의 많은 관심을 받았으며, 미표준국인 NIST는 2001년에 이를 보완하여 FIPS 186-2를 공표하게 된다.

인터넷의 보급 초기에 널리 사용되었던 인터넷 브라우저 프로그램 네비게이터로 우리에게 친숙한 네스케이프사의 난수발생기에 대한 안전성 분석은 2001년도에 Goldverg와 Wagner[2]에 의해서 이루어졌다. 이들은 당시 네스케이프사의 난수발생기에서 사용되는 종자(seed) 값은 시간 정보와 ID 관련 정보만으로 구성되어 매우 취약하다는 사실을 발견하였다. 이러한 약점은 전자 상거래 등을 위한 정보보호 시스템인 SSL을 지원하는 네스케이프 웹 브라우저의 결정적인 결함이 되었다.

위의 사례들로 비추어 볼 때, 난수발생기는 대부분의 암호 시스템에서 필수불가결한 요소일 뿐만 아니라 안전성 측면에서도 매우 중요하다는 것을 알 수 있다. 본 논문에서는 먼저 암호 시스템에 사용되는 난수발생기를 하드웨어 기반과 소프트웨어 기반으로 대별하여 현황을 파악한다. 하드웨어를 기반으로 하는 난수발생기의 종류 및 사용 실태와 안전성 분석 기술을 조사 분

* 국민대학교 자연과학대학 수학과 (jskang@kookmin.ac.kr)

석하며, 소프트웨어 기반 난수발생기에 대한 최신 설계 기술을 정리해 본다. 그리고 난수발생기의 출력 수열에 대한 통계적 안전성 평가 방법과 난수발생기 알고리즘에 대한 이론적 안전성 분석 기술에 대해서 논한다.

Ⅱ. 난수발생기 개요

암호 시스템에서 사용하는 난수발생기는 자연 현상의 랜덤성을 이용하여 참다운 의미의 난수를 생성하는 RNG(random number generator)와 난수처럼 보이는 의사 난수를 생성하는 PRNG(pseudo-random number generator)로 대별할 수 있다. RNG는 엔트로피 소스라 불리는 비결정적인(non-deterministic) 소스를 사용하여 랜덤한 비트 수열을 생성한다. RNG의 출력 수열은 고유의 편차(bias)를 갖기 때문에 적절한 보정을 통해서 랜덤한 수열로 사용되는 것이 일반적이며, PRNG의 초기 입력값으로 사용되기도 한다.

PRNG는 종자(seed)라 불리는 입력값을 사용하여 RNG보다는 매우 효율적인 방법에 의해서 난수열을 생성한다. PRNG는 소프트웨어 프로그램으로 구현되는 것이 일반적이며, 암호 시스템 내에서 대칭키나 공개키 암호 알고리즘들과 함께 중요한 안전성적 요소로 인식되고 있다.

이론적으로 난수성(randomness)은 확률론에서 균일한 동전을 연속적으로 던지는 독립 시행과 대응하는 비트 수열을 생성하는 것을 기준으로 평가한다. 즉, 0과 1이 나타날 확률이 1/2로 동일하고, 이들의 연속적인 확률 분포가 독립적이고(independent), 동일한 분포(identically distributed)를 하는 확률변수열(sequence of random variables)이라는 사실이 난수성을 만족하는 것으로 정의된다.

난수발생기의 출력 수열 관점에서만 난수성을 평가한다면 위의 확률론적인 정의만으로 충분하다. 그러나 우리가 암호 시스템에서 사용하는 난수발생기는 하나의 장치이기 때문에 어떻게 사용되느냐에 따라서 안전성이 영향을 받게 된다. 이를테면, PRNG를 사용할 경우에 종자값을 비밀로 하면 이전 출력값을 안다고 할지라도 다음 출력값을 예측할 수 없어야 한다. 이러한 성질을 순방향 비예측성(forward unpredictability)이라 하는데, 이를 보장하기 위해서 종자값은 반드시 비밀로 간직해야 하는 것이다.

난수발생기는 각종 암호시스템에서 없어서는 안될 중요한 요소이기 때문에 여러 표준화 그룹과 정보보호 업체들에서 이에 대한 요구조건들을 정립해 놓았다. 미국의 RSA사에서는 소프트웨어적 방법으로 난수를 생성할 경우의 권고사항을 발표하였으며[3], 공개키 암호 관련 표준화 그룹인 IEEE P1363[4]에서는 일반적인 범용의 난수발생기에 대한 암호적 난수발생기의 차이를 인식시키고 이에 대한 몇 가지 기준들을 제시하였다. 최근에는 인터넷 표준화 그룹인 IETF에서 RFC 1750의 개선된 버전으로 RFC 4086[5]에서 보안을 위한 난수성의 요구조건을 정립하고, 인터넷을 비롯한 네트워크 보안 통신에 사용 가능한 PRNG들을 소개하였다. 또한, 미국의 표준국인 NIST에서는 난수발생기도 블록 암호(block cipher), 스트림 암호(stream cipher), 또는 공개키 암호 알고리즘과 같은 중요한 암호 알고리즘 중의 하나로 인식하여 PRNG를 이용한 난수 발생 기법에 대한 권고를 발간하였다[6].

실제로 사용되는 암호시스템에서 RNG는 하드웨어 기반으로 만들어진 장치 또는 기기이며, PRNG는 소프트웨어 기반으로 설계된 알고리즘을 구현한 프로그램의 형태를 띠고 있다. 여기에서는 이를 분리해서 조사 분석해보도록 한다.

Ⅲ. 하드웨어 기반 RNG

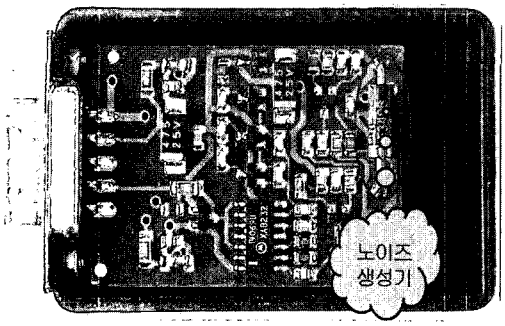
하드웨어 기반 난수발생기는 본질적인(genuine) 난수를 생성하는 전자적 장치(electronic device)로 컴퓨터에 연결하여 사용된다. 하드웨어 기반 RNG는 컴퓨터 프로그램에 의해서 생성되는 의사난수와는 구별되는 참다운 의미의 난수를 생성하는 것이다[7].

하드웨어 RNG는 상업적인 용도로 많이 이용된다. 복권 등에서의 당첨번호 추첨에서부터 의학, 생물학, 심리학 등에서의 실험계획, 통계적 시뮬레이션 등 다양한 용도를 갖는다. 암호시스템에서는 가장 중요한 비밀 요소 중의 하나인 암호화를 위한 키를 생성하기 위해서 이 하드웨어 RNG가 사용된다.

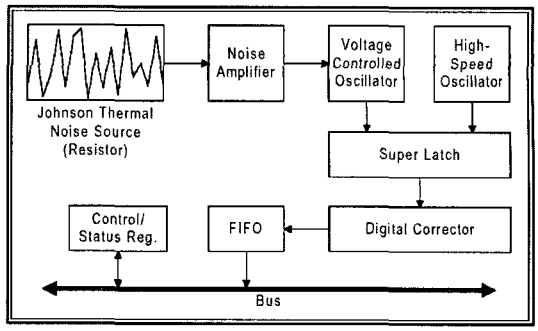
하드웨어 RNG는 레지스터 또는 반도체로부터 발생하는 전자적 노이즈(electrical noise)와 방사능 물질의 붕괴 시간(decay times)을 난수성의 원천으로 사용하는 것이 대부분이다.

1. 전자적 노이즈를 이용한 하드웨어 RNG

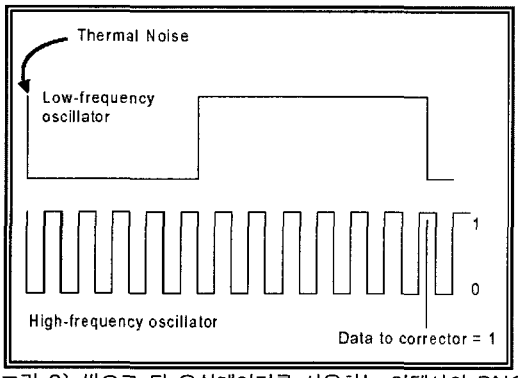
난수성을 채집하는 원천으로 반도체나 레지스터에서 발생하는 전자적 노이즈를 이용한 하드웨어 RNG



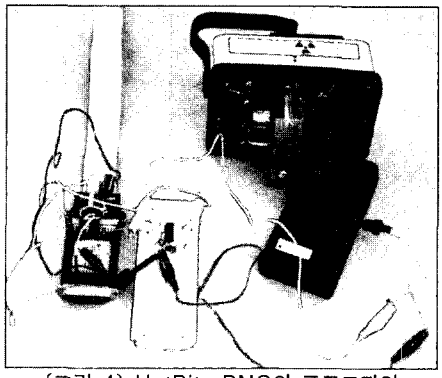
〔그림 1〕 스웨덴 Protego사의 하드웨어 RNG



〔그림 2〕 인텔사의 RNG



〔그림 3〕 쌍으로 된 오실레이터를 사용하는 인텔사의 RNG



〔그림 4〕 HotBits RNG의 프로토타입

가 가장 많다고 볼 수 있다. 그 대표적인 예는 스웨덴의 Protego사의 제품에서 찾아볼 수 있는 것으로 그림 1에 나타나 있다.

상업용으로 나와 있는 Protego사의 RNG는 위 그림에서 보는 바와 같이 시리얼 포트를 이용하여 컴퓨터와 플러그인 된다. 그림 우측에 위치한 다이오드가 노이즈 생성기가 되고, 나머지 회로들은 증폭기와 노이즈로부터 난수열을 가공하는 장치들이다.

노이즈 소스로 레지스터에서 발생하는 열을 이용하는 대표적인 예는 인텔사의 RNG에서 볼 수 있다[8]. 인텔의 RNG는 "존슨 열 노이즈(Johnson Thermal Noise)"라 불리는 열 노이즈를 이용하여 난수를 생성한다. 인텔의 RNG에 대한 블록 다이어그램은 그림 2에 나타나 있다.

인텔사의 RNG는 열 노이즈를 이용하고, 두 개의 오실레이터(oscillator)를 쌍으로 이용하여 초기의 난수를 생성한다. 두 개의 오실레이터 중 하나는 빠르게 진동하고, 다른 하나는 느리게 진동한다. 열 노이즈 소스는 느린 클럭의 주기를 갖는 오실레이터의 입력으로 사용되는 것이다. 쌍으로된 오실레이터의 구성은 그림 3에 나타나 있다.

2. 방사능 붕괴 시간을 이용한 하드웨어 RNG

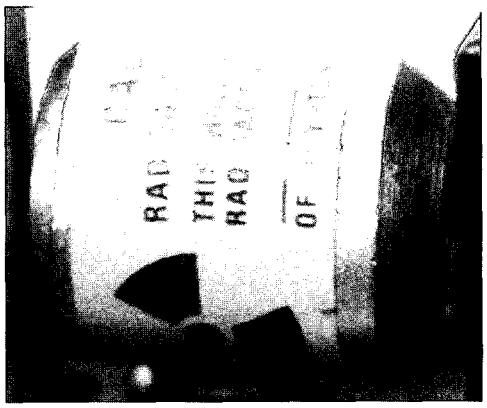
방사성 물질의 붕괴 시간(decay times)을 이용한 하드웨어 RNG의 대표적인 예는 스위스 Autodesk사에서 제작한 HotBits RNG[9]를 들 수 있다. HotBits RNG에 대한 프로토 타입은 그림 4에 나타나 있다.

한편, 방사능 물질은 우리가 살고 있는 주변 환경 어디에서나 관측 가능하다. 그러나 일상적인 방사능 물질을 이용하는 것은 수 많은 비트의 난수를 생성하기 위해서 효율적이지 못하다. 지질의 연대를 측정하기 위해서 주로 사용되는 우라늄이나 토륨 등의 방사성 동위원소는 반감기가 너무 길기 때문에 실질적인 붕괴 시간을 측정하는 것이 매우 어려울 수 있다. 난수발생기를 위한 원천으로 사용하기에는 일상적인 방사능 물질보다 훨씬 강력한 의미의 방출 소스가 필요하다.

HotBits RNG에서 사용한 방사성 동위원소는 Krypton-85이다. Krypton-85는 반감기가 10.73년으로 알려져 있으며, 다음과 같은 베타 붕괴를 통해서 랜덤성의 원천으로 사용된다.

$$^{85}Kr \rightarrow ^{85}Rb + \beta^{-} + \gamma$$

HotBits RNG에서 Krypton-85의 붕괴 시간을 랜덤성의 원천으로 한 소스 캡슐이 그림 5에 나타나 있다.



〔그림 5〕 Krypton-85를 이용한 HotBits RNG의 방사능 붕괴 시간 소스 캡슐

3. 하드웨어 RNG의 보정기(corrector)

일반적으로 하드웨어 RNG에서 사용하는 노이즈 소스는 장치 고유의 성질에 따른 편차를 가지기 때문에 그 출력 수열은 균일 랜덤성(uniform randomness)으로부터 차이가 있는 분포를 가질 수 있다. 이러한 편차의 원인이 되는 요인들을 열거하면 다음과 같다.

- 편차(bias)
- 치우침 현상(drift)
- 단기 자기 상관성(short term autocorrelation)
- 단기 종속성(short term dependencies)
- 이산 주기(discrete frequencies)
- $1/f$ 노이즈
- 돌발 멈춤(bad spot)
- 백 도어(back door)

하드웨어 RNG의 초기 출력(raw sequence)은 장치 특성에 따라 위와 같은 편차들을 포함하고 있기 때문에 반드시 보정기(corrector)를 사용하여 균일 분포를 이루도록 만들어 주어야 한다. 보정기로 가장 많이 사용되는 것은 XOR 보정기와 Von Neumann 보정기이다.

XOR 보정기는 원래의 출력 수열을 쌍으로 묶어 비트별 XOR 연산을 하여 결과를 난수로 사용하는 방식이다. 그리고 Von Neumann 보정기는 원래의 출력 수열을 쌍으로 묶는 것은 XOR 보정기와 같지만 두 비트가 동일할 경우에는 그 것들을 버리고 다른 경우에는 첫 번째 비트를 최종 출력 수열을 난수로 택하는 방법이다. 이 두 보정기에 대한 예는 다음 그림 6에 나타나 있다.

```
Raw
10110010100100011010010111001000101001011 1
XOR
1 0 0 1 1 1 0 1 1 1 1 1 0 0 1 0 1 1 1 1 0
Von Neumann
1     1 1 0   0 1 1 0 0     1   1 1 0 0
```

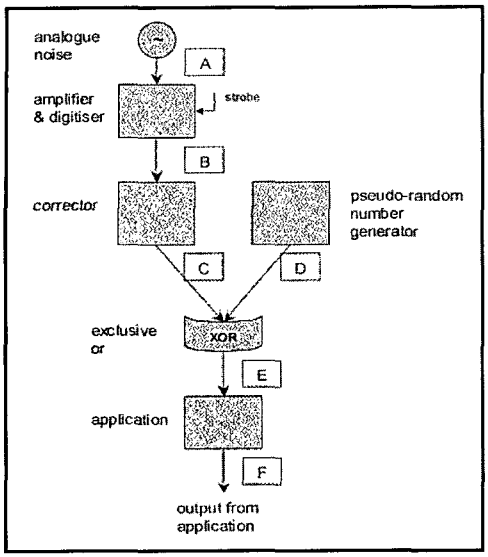
〔그림 6〕 XOR 보정과 Von Neumann 보정 방법

보정기는 아날로그 노이즈 소스가 갖는 다양한 편차들을 없애주는 역할을 하며, 위에서 언급한 효율적인 방법들인 XOR 보정 방법과 Von Neumann 보정 방법을 사용할 경우 보정 효과가 매우 높은 것으로 알려져 있다[7].

4. 하드웨어 RNG의 안전성 평가

하드웨어 RNG는 소프트웨어 PRNG와는 여러모로 다르기 때문에 이에 대한 안전성 평가 방법도 달라야 한다는 것이 합리적인 생각이다[7]. 물론 최종 출력 수열만을 고려하여 통계적으로 난수성을 평가하는 방법에서 큰 차이가 날 이유는 그다지 많지 않다. 다만, 하드웨어 RNG는 기계적으로 여러 가지 구성 요소들로 이루어져 있기 때문에 안전성 평가를 어느 지점에서 실시하느냐가 중요한 요인이 된다. 다음 그림 7은 하드웨어 RNG에서 안전성 평가 지점(test point)을 여러 개 택할 수 있음을 보여주고 있다.

위 그림에서 A 지점은 일반적으로 사용자들이 검사할 수 있는 지점이 아니다. 이 지점에 대해서는 제조자가 몇 가지 정보를 제공할 수 있다. 하드웨어 RNG의 안전성 평가에서 가장 중요한 단계는 B 지점에서의 행태를 관찰하는 것이다. 보정기를 통과하기 전에 하드웨어 RNG의 특성을 제대로 담고 있는 수열이 바로 B 지점이기 때문이다.



〔그림 7〕 하드웨어 RNG에서의 안전성 평가 지점

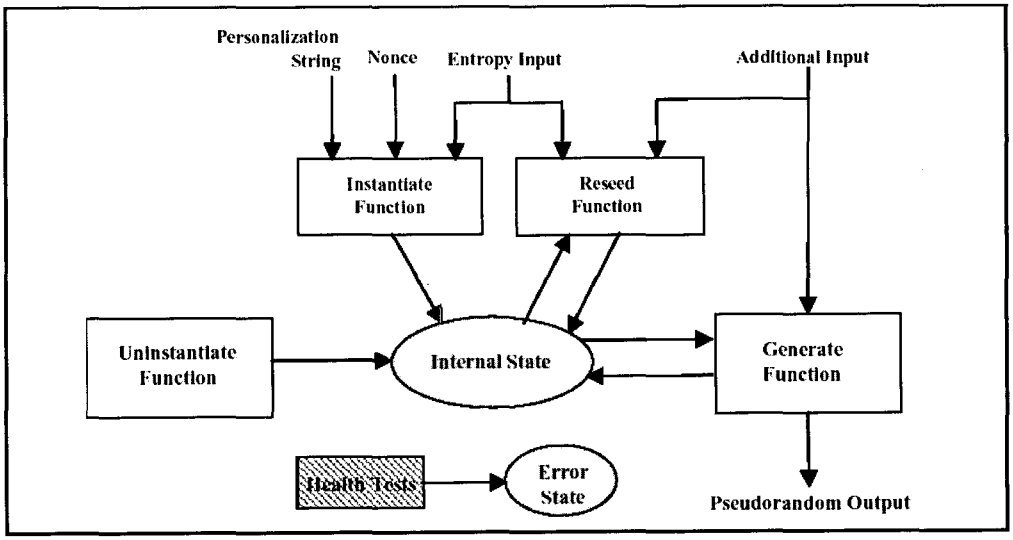
보정기를 통과한 후의 수열을 관찰할 수 있는 C 지점에서의 평가는 B 지점에서의 평가 결과와 함께 행해진다면 보정기의 성능을 평가한다는 의미가 있다. 그러나 C 지점에서의 평가만 이루어진다면 하드웨어 RNG의 특성이 무시될 수 있기 때문에 제한적인 평가가 이루어질 뿐이다. D, E, F 지점들에서의 안전성 평가도 상호 비교해 가면서 이루어질 경우에 중요한 의미를 가진다. 그러나 이들 지점에서의 안전성 평가는 하드웨어 RNG에만 국한된 것으로 보기는 어렵다. 난수발생기의 출력 수열을 통계적으로 평가하는 작업과 크게 달라질 것은 없어 보이기 때문이다.

Ⅳ. 소프트웨어 기반 PRNG

미국 표준국 NIST(National Institute of Standards and Technology)는 최근에 암호 기술이 응용되는 곳에서 난수가 필요할 경우에 이를 생성하는 기술을 표준화하기 위한 권고 문서인 SP 800-90[6]을 발간하였다. 이 문서에서는 하드웨어 기반 RNG를 예측불능인 물리적 과정을 이용하기 때문에 "비결정적 랜덤 비트 생성기(Non-deterministic Random Bit Generator, 이하 NRBG)"라 하고, 결정적인 알고리즘에 의해서 난수를 생성하는 소프트웨어 기반 난수발생기를 "결정적 랜덤 비트 생성기(Deterministic Random Bit Generator, 이하 DRBG)"라 명명하였다. 그리고 지금까지의 DRBG 관련 기술을 분석 종합하여 미 연방정부의 표준안으로 승인하기 위한 DRBG 메커니즘을 기술하였다. SP 800-90 문서가 소프트웨어 기반 의사난수발생기의 최신 기술에 대한 조사 분석 내용을 충실히 담고 있기 때문에 이를 중심으로 살펴보기로 한다.

1. DRBG의 구성 요소

그림 8은 DRBG의 기능적 모델을 보여주고 있다.



〔그림 8〕 DRBG의 기능적 모델

DRBG 알고리즘 구성 요소의 기능에 따른 분류는 엔트로피 입력, 내부 상태(internal state), 구동 함수(instantiate function), 종자 재입력 함수(reseed function), 생성 함수(generate function) 등을 포함한다.

1.1. 종자(seeds)

종자는 DRBG 알고리즘의 초기 입력값 역할을 한다. 종자의 입력으로 DRBG는 작동하게 되고 반복적인 메커니즘에 의해서 원하는 난수열을 생성하는 것이다. 그러므로 종자를 어떻게 생성할 것인가의 문제가 안전성적으로 중요하다. 일반적인 DRBG에서 종자의 생성은 엔트로피 입력(entropy input), 논스(nonce), 개별화 특성값(personalization string) 등이 사용된다. 개별화 특성값은 응용 환경에 따라 선택적으로 적용된다. 그림 9는 DRBG 구동을 위한 종자의 구축과정을 보여주고 있다. 그림에서 "Opt df"는 종자 생성 함수를 선택적으로 설계할 수 있음을 의미한다.

종자의 구축 과정에서 논스(nonce)는 어떤 공격에 대한 안전성을 제공하기 위해서 권고되는 요소이다. 논스에는 적어도 원하는 엔트로피의 1/2을 갖는 랜덤 수 또는 반복주기가 1/2 security-strength 비트를 넘는 랜덤하지 않은 수가 사용 가능하다. 엔트로피 입력과 같은 소스를 사용할 경우에는 엔트로피 입력 부분의 전체 엔트로피가 3/2 security-strength 비트보다 커야 한다. 논스를 사용하지 않고 DRBG를 여러 번 사용할 경우 안전성이 저하되는 결과를 초래할 수 있다.

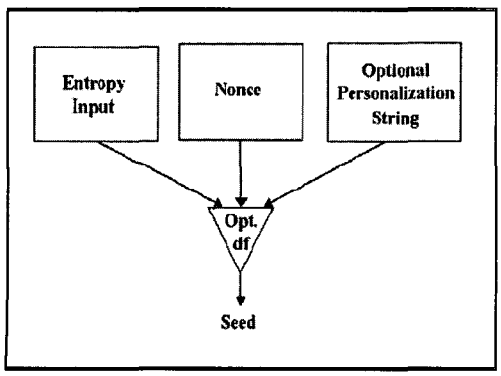
개별화를 위한 특성값(personalization string)은 DRBG의 유일성(uniqueness) 보장이 필요한 경우에 사용된다. 개별화 특성값으로 사용 가능한 것들을 다음과 같다.

- 장치의 시리얼 넘버(device serial number)
- 사용자 식별자(user identification)
- 공개키(public key)
- 개인키(private key)
- 개인 식별자(PIN) 및 패스워드(password)
- 타임스탬프(timestamp)
- 네트워크 주소(network address)
- 응용 식별자(application identifier)
- 프로토콜 버전 식별자(protocol version identifier)

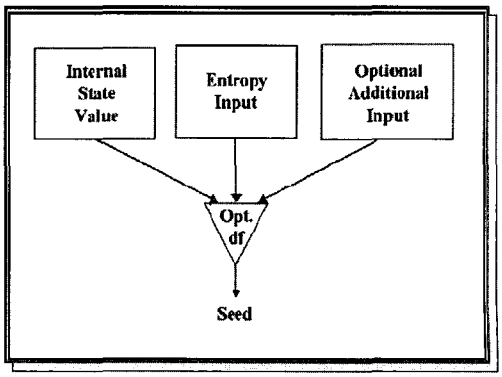
한편, 한 개의 종자로부터 너무 많은 출력 수열을 뽑아내게 되면, 예측 공격에 대한 저항성이 떨어질 수 있다. 이러한 것에 대비하기 위한 조치가 종자 재입력 과정(reseeding)이다. DRBG 작동 시 종자 재입력 과정은 주기적으로 이루어지며 엔트로피 입력, 내부 상태값, 그리고 선택적으로 추가적인 정보를 사용하여 이루어진다. 그림 10은 종자 재입력 과정을 나타내고 있다.

1.2. 엔트로피 입력(entropy input)

DRBG 알고리즘을 구동시키기 위한 초기 입력은 종자이지만, 이 종자 값을 구축하기 위해서 가장 중요한 요소가 바로 엔트로피 입력이다. 엔트로피 입력은 랜덤성의 원천이기 때문에 하드웨어 RNG의 출력값을 사용하는 것이 적절하다. 그러나 비용과 효율성 문제 때문에 하드웨어 RNG를 사용할 수 없는 환경에서는 소프트웨어적으로 습득 가능한 엔트로피 입력을 사용해야 한다.

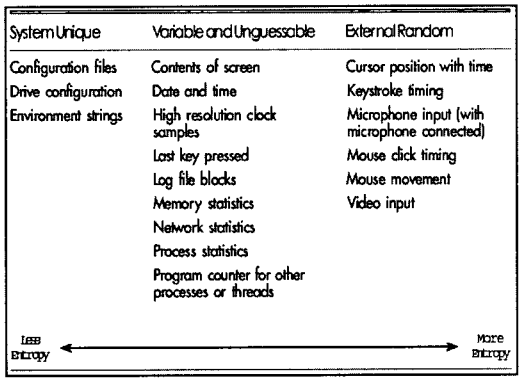


〔그림 9〕 종자(seed)의 구축 과정



〔그림 10〕 종자 재입력(reseeding) 과정

소프트웨어적으로 엔트로피 입력을 습득하는 것에는 화면, 시스템 클럭, 프로그램 카운터 등의 시스템 정보, 범용 컴퓨터 외부 주변 장치로부터 입력 받는 정보 또는 사용자로부터 받는 키보드 입력값, 마우스 좌표 값의 변이 정보 등을 이용하는 방법이 있다. 이들 정보의 엔트로피는 예측 불가능성과 사용되는 데이터의 양에 따라서 결정된다. 예측이 쉬운 정보라 할지라도 많은 양의 데이터를 사용하면 엔트로피는 높아질 수 있는 것이다. 그림 11은 예측이 어려운 정도에 따라서 분류한 소프트웨어적인 엔트로피 입력을 나타낸 것으로 RSA사[3]에서 분류한 것이다.



〔그림 11〕 소프트웨어적인 엔트로피 입력

2. NIST의 DRBG 알고리즘 규격

NIST는 암호시스템에서 이용 가능한 알고리즘들과 난수발생기의 안전성을 고려하여 몇 가지 형태의 DRBG를 선정하여 규격화하였다. 규격화된 알고리즘들은 사용되는 암호 알고리즘에 따라서 해쉬 함수 기반 DRBG, 블록 암호 기반 DRBG, 공개키 암호 기반 DRBG 등의 형태로 나타난다.

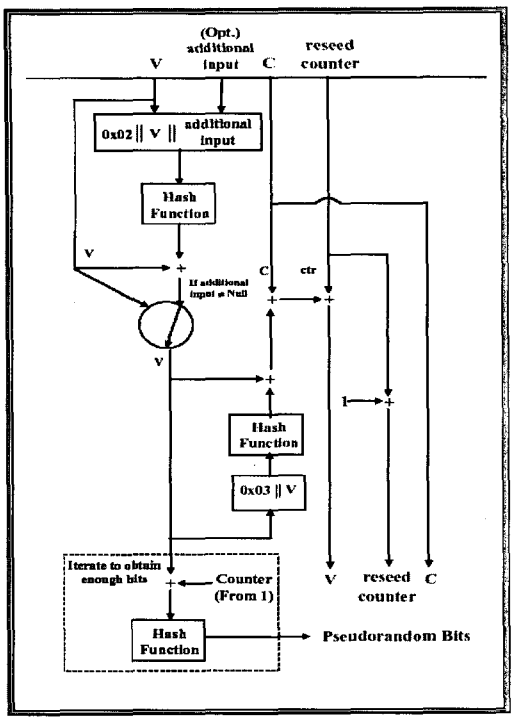
2.1. 해쉬 함수 기반 DRBG

DRBG는 일방향 또는 비가역적(non-invertible)인 해쉬 함수를 기반으로 설계될 수 있다. NIST가 규격화한 해쉬 함수 기반 DRBG는 "Hash_DRBG"와 "HMAC_DRBG"가 있으며, 사용되는 해쉬 함수는 SHA 계열이다. 해쉬 함수 기반 DRBG의 구체적인 파라미터는 표 1과 같다.

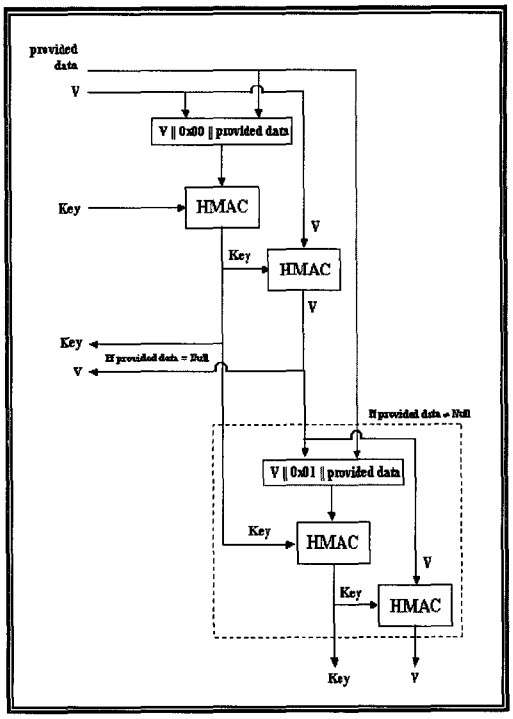
Hash_DRBG에서는 구동(instantiate), 종자 재입력, 생성 함수로 하나의 해쉬 함수를 사용한다. Hash_DRBG의 블록도는 그림 12에 나타나 있다. 그림에서 V는 종자와 같은 비트 길이로 DRBG를 부를 때마다 업데이트된다. 그리고 C는 종자에 의존하는 상수값으로 종자와 같은 비트 길이를 갖는다. 이 두 개의 V와 C 값이 내부 상태의 비밀 값이 되어 DRBG의 안전성에 결정적인 값들이다. 종자 재입력을 위한 카

〔표 1〕 해쉬 함수 기반 DRBG를 위한 파라미터

	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512
Supported security strengths	See SP 800-57				
highest_supported_security_strength	See SP 800-57				
Output Block Length (*outlen*)	160	224	256	384	512
Required minimum entropy for instantiate and reseed	*security_strength*				
Minimum entropy input length (*min_length*)	*security_strength*				
Maximum entropy input length (*max_ length*)	$\leq 2^{35}$ bits				
Seed length (*seedlen*) for Hash_DRBG	440	440	440	888	888
Maximum personalization string length (*max_personalization_string_length*)	$\leq 2^{35}$ bits				
Maximum additional_input length (*max_additional_input_length*)	$\leq 2^{35}$ bits				
max_number_of_bits_per_request	$\leq 2^{19}$ bits				
Number of requests between reseeds (*reseed_interval*)	$\leq 2^{48}$				



〔그림 12〕 Hash_DRBG 블록도



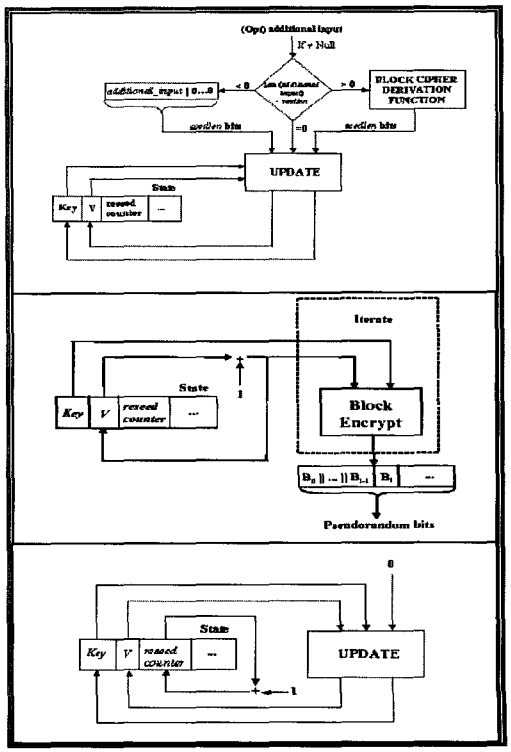
〔그림 13〕 HMAC_DRBG 업데이트 함수

운터(counter)는 요구되는 의사 랜덤 비트 수를 나타낸 것으로 새로운 엔트로피 입력을 얻기 위해서 사용되는 것이다.

HMAC_DRBG는 일반적인 해쉬 함수 대신 키가 있는 해쉬 함수(keyed hash function)를 사용하는 방식이다. 그림 13은 HMAC_DRBG의 업데이트 함수를 나타낸 것이다. 업데이트 과정에서 HMAC이라는 표준화된 MAC 알고리즘을 사용하고, 키가 입력값으로 사용된다는 점이 Hash_DRBG와 구별되는 점이다.

2.2. 블록 암호 기반 DRBG

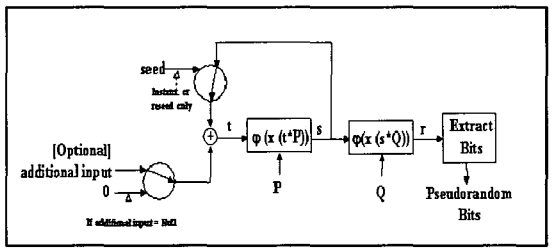
블록 암호 기반 DRBG는 핵심 함수로 DES 또는 AES 등과 같은 블록 암호 알고리즘을 사용하는 의사 난수 생성법이다. SP 800-90 문서에서는 블록 암호 운영 모드 중에서 카운터 모드를 응용한 CTR_DRBG를 규격화 하고 있으며, 사용되는 알고리즘과 키 길이는 3-Key-TDES, AES-128, AES-192, AES-256 이다. CTR_DRBG의 블록도는 그림 14에 나타나 있다.



〔그림 14〕 CTR-DRBG 블록도

2.3. 공개키 암호 기반 DRBG

DES 또는 AES와 같은 대칭키 암호 알고리즘에 비해서 RSA나 타원곡서 암호 같은 공개키 암호 알고리즘은 속도가 매우 느리기 때문에 의사난수 생성기의 핵심 함수로 사용하기에는 부적합하다. 그러나 응용 환경에 따라서 해쉬 함수나 대칭키 암호 알고리즘 같은 효율성 높은 알고리즘을 이용할 수 없는 경우가 있다. 이러한 경우에는 비효율적이지만 공개키 암호 시스템만을 이용한 의사난수 생성법이 필요하다. 이러한 경우를 대비하기 위해서 NIST는 타원곡선 이산대수 문제에 기반한 DRBG를 규격화하였다. 규격화된 의사난수 생성법은 Dual_EC_DRBG이며, 이는 타원곡선 위의 두 점 P, Q가 주어졌을 때, $Q=aP$를 만족하는 a를 찾는 문제인 ECDLP에 기반한 것이다.



〔그림 15〕 Dual-EC-DRBG 블록도

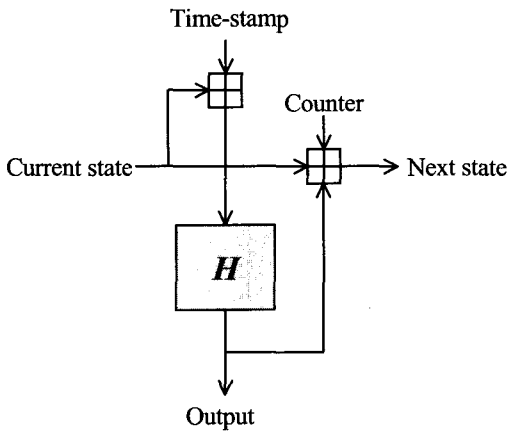
3. FIPS 186 및 ANSI X9.17 PRNG

FIPS 186 PRNG[10]는 미국 연방 표준 디지털 서명 알고리즘인 DSA에서 필요한 난수를 생성하기 위한 목적으로 제정된 표준 의사난수발생기이다. 해쉬 함수 기반이며, DSA 알고리즘의 개인키 및 메시지별 비밀 값을 생성하기 위해서 사용된다. 이 FIPS 186 PRNG는 앞에서 소개한 바와 같이 안전성적 결함이 발견되어 2001년도에 FIPS 186-2[1]라는 수정된 버전이 발표되었다. 전체적인 구조에는 변함이 없고 세부적인 알고리즘에서 벡터의 길이를 두 배로 증가시켜서 원하는 엔트로피를 확보하는 방향으로 수정된 것이다. FIPS 186 PRNG의 기본 구조는 그림 16에 나타나 있다.

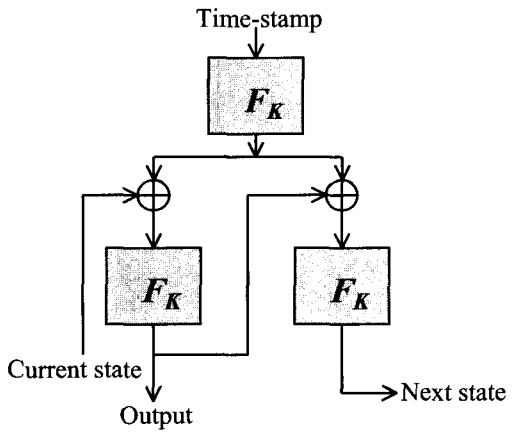
ANSI X9.17 PRNG[11]는 미국 표준으로 은행 등의 금융권에서 사용할 목적으로 제정된 의사난수발생기이다. 구체적으로는 블록 암호 DES의 키와 초기값(IV)을 생성하는 데에 사용된다. ANSI X9.17 PRNG의 기본 구조는 그림 17에 나타나 있으며, 그림에서 F_K는 두 개의 키를 갖는 E-D-E 모드의 삼중 DES를 가리킨다.

4. BSAFE PRNG

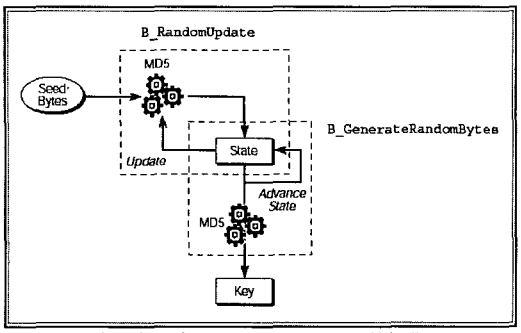
RSA사의 제품인 BSAFE[12]에서의 의사난수 생성은 Intel RNG[8]를 이용하여 종자 값을 얻고, MD5라는 해쉬 함수를 이용하여 내부 상태를 업데이트 하는 방식으로 의사난수를 생성한다. BSAFE PRNG의 기본 구조는 그림 18에 나타나 있다.



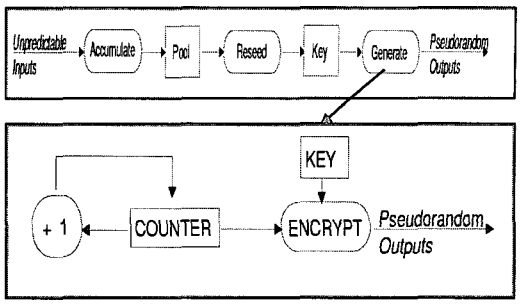
〔그림 16〕 FIPS 186 PRNG 기본 구조



〔그림 17〕 ANSI X9.17 PRNG 기본 구조



〔그림 18〕 BSAFE PRNG 기본 구조



〔그림 19〕 Yarrow-160 PRNG

5. Yarrow-160 PRNG

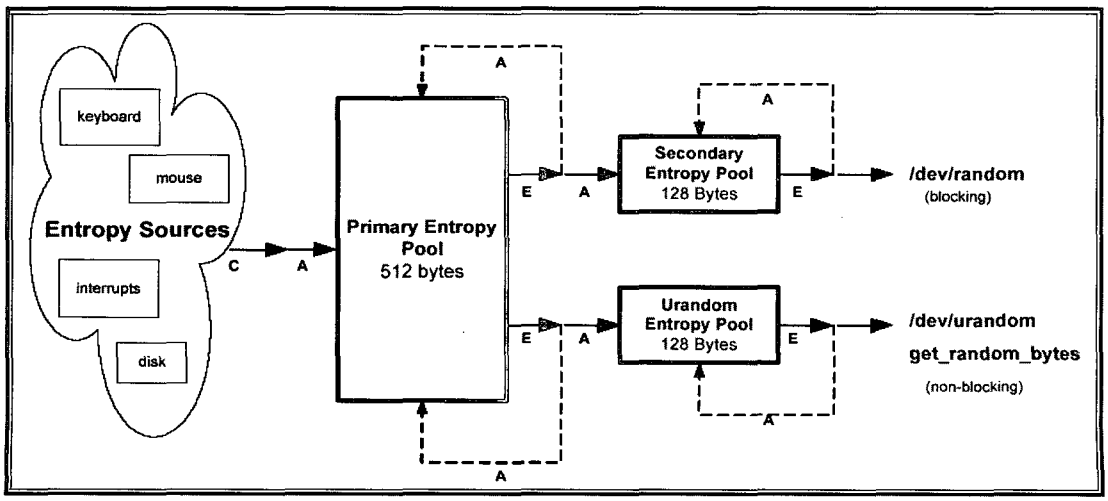
Yarrow-160 PRNG[13]는 미국의 Counterpane 사에서 발표한 의사난수 생성기로 블록 암호 알고리즘을 기반으로 한 CTR_DRBG의 효시가 된 것으로 볼 수 있다. NIST의 SP 800-90[6] 문서에서 논하고 있는 DRBG의 구성 요소들을 처음으로 정립한 것도 Yarrow-160을 개발한 Counterpane사였다는 점에서도 역사적 의미가 있는 PRNG로 생각된다. Yarrow-160에서 정립한 의사난수 생성 과정과 생성 함수는 그림 19에 나타나 있다.

6. Linux PRNG

컴퓨터 운영체제 중의 하나인 Linux 시스템에서 사용되고 있는 의사난수 발생기는 최근에 역공학 (reverse engineering) 기법에 의해서 소스 코드가 알려지게 되었다. 지난 5월에 열린 IEEE SSP2006 학술대회에서 Gutterman과 Pinkas[14]는 Linux PRNG의 정확한 구조를 밝히고, 이에 대한 안전성 분석을 실시한 논문을 발표하였다. 그들에 의하면 Linux PRNG의 순방향 안전성(forward-security)을 2^{64}의 공격량으로 분석 가능하다는 것이다. 이를 효시로 하여 앞으로 Linux PRNG에 대한 안전성 분석 관련 연구가 활발해질 것으로 보인다. Linux PRNG의 기본 구조는 그림 20에 나타나 있다.

7. 기타의 PRNG

지금까지 열거한 소프트웨어 기반 PRNG 외에도 의사난수를 생성하기 위한 다양한 방법들이 존재한다. SSL과 TLS[15] 시스템에서는 HMAC 함수를 핵심



〔그림 20〕 Linux PRNG의 기본 구조

함수로 사용한 PRNG를 표준으로 정하고 있다. SSL과 TLS에서 규격화된 PRNG는 다음과 같다.

$$out(0) = HMAC(key, HMAC(key, seed) \| seed)$$
$$out(n) = HMAC(key, out(n-1) \| seed)$$

TLS 시스템에서는 HMAC-MD5와 HMAC-SHA1 기반 PRNG의 출력을 비트별 XOR 연산한 결과를 의사난수로 사용하는데, 이는 둘 중 어느 하나의 알고리즘의 안전성이 문제가 될지라도 전체적인 PRNG의 안전성에는 그 영향을 최소화하도록 하는 조치이다.

SSH 시스템[16]에서도 다음과 같은 해쉬 함수 기반 PRNG를 사용하고 있다.

$$out(0) = hash(key \| exchange\ hash \| session\ ID)$$
$$out(n) = hash(key \| exchange\ hash \| out(n-1))$$

그밖에도 IPsec, PGP, S/MIME 등의 시스템에서 해쉬 함수 기반 PRNG를 사용하고 있다.

V. 출력 수열에 대한 통계적 안전성 평가

RNG 및 PRNG에 대한 안전성 평가는 크게 두 가지 관점으로 분류할 수 있다. 첫째는 RNG 및 PRNG의 출력 수열을 확률론을 바탕으로 통계적인 방법에 의해서 평가하는 것이다. 이는 난수발생기 자체에 대한 평가라기 보다는 출력 수열의 통계적 특성만을 고려하는 것이므로 암호 알고리즘 설계 이론과는 무관하다고 할 수 있다. 둘째는 RNG 및 PRNG를 알고리즘 설계 관점에서 안전성을 평가하는 방법이다. 이 두 번째 방법은 다음 장에서 논하기로 한다.

통계적인 방법으로 출력 수열의 난수성을 평가하는 방법은 여러 가지가 있다. 원시적인 방법 중의 하나로 Golomb[17]의 pn-수열(pseudo-noise sequence)을 들 수 있다. pn-수열은 의사랜덤 수열이 갖추어야 할 조건을 세 가지로 정하고 이를 만족하는 수열을 말하며, 이 조건 중 하나라도 어길 경우 랜덤하지 않은 수열로 판정하는 방법이다.

FIPS 140-1[18]는 암호 시스템의 보안 요구 조건 중의 하나로 전원을 연결할 때에 다음 네 가지 난수성 검사를 기본적으로 실시하도록 하였다. 이 때 사용되는 출력 수열의 길이는 20,000 비트이다.

- 단일 비트 검정(monobit test)
- 포커 검정(poker test)
- 런 검정(runs test)
- 롱런 검정(long run test)

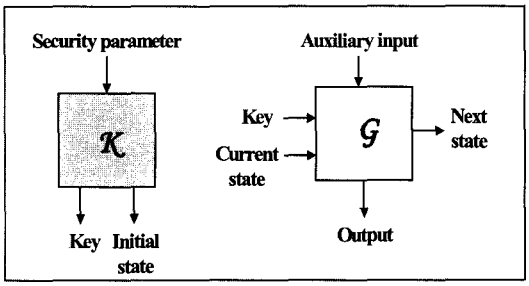
정형화 되어 있지 못하던 통계적 난수성 검정법이 규격화된 것은 NIST가 AES 프로젝트를 추진하는 과정에서 개발한 통계적 검정 도구[19] 때문이다. AES 프로젝트 후보 알고리즘들을 평가하는 과정에서 중요한 기준으로 사용되었던 이 통계적 검정 도구는 그 이후로 블록 암호 알고리즘뿐만 아니라 각종 난수발생기의 출력 수열에 대한 통계적 평가 도구로 이용되고 있다. NIST의 통계적 검정 도구에 포함된 16가지 검사 항목은 다음과 같다.

1. The Frequency (Monobit) Test,
2. Frequency Test within a Block,
3. The Runs Test,
4. Test for the Longest-Run-of-Ones in a Block,
5. The Binary Matrix Rank Test,
6. The Discrete Fourier Transform (Spectral) Test,
7. The Non-overlapping Template Matching Test,
8. The Overlapping Template Matching Test,
9. Maurer's "Universal Statistical" Test,
10. The Lempel-Ziv Compression Test,
11. The Linear Complexity Test,
12. The Serial Test,
13. The Approximate Entropy Test,
14. The Cumulative Sums (Cusums) Test,
15. The Random Excursions Test, and
16. The Random Excursions Variant Test.

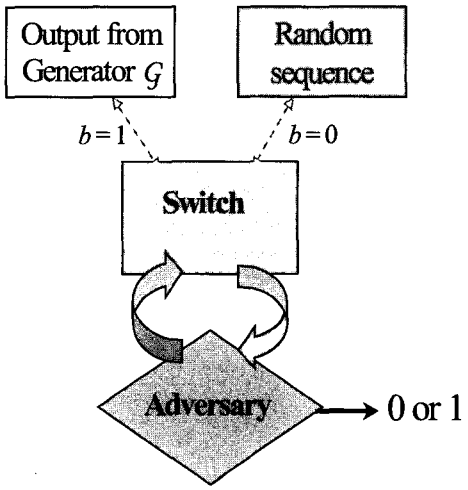
VI. 난수발생기에 대한 이론적 안전성 분석

그동안 난수발생기에 대한 안전성 평가는 출력 수열에 대한 통계적 평가가 주류를 이루었다. 그러나 최근에는 난수발생기도 블록 암호나 해쉬 함수와 같은 암호 알고리즘의 설계 관점에서 내부 구성 함수들의 안전성을 분석하는 연구가 진행되고 있다. 증명 가능 안전성(provable security) 관점에서 실용적인 PRNG의 안전성을 분석하고자 하는 이 안전성 분석법은 Eurocrypt 2002 학술대회에서 Desai-Hevia-Yin[20]에 의해서 처음으로 제기되었다. 이들은 PRNG 알고리즘을 그림 21과 같이 단순화 시킨 다음에 비밀로 하는 요소가 어떤 것이냐에 따라 공격

을 정의하였다. 그리고 본질적인 랜덤(truly random) 수열과 PRNG의 출력 수열을 구별할 수 있는 확률값에 의해서 그 안전성을 분석하였다. 구별 가능성을 분석하는 메커니즘은 그림 22에 나타나 있다.



〔그림 21〕 이론적 안전성 분석을 위한 PRNG 모델



〔그림 22〕 PRNG의 구별 가능성 분석 메커니즘

이와 같은 이론적 안전성 분석 방법에 의해서 Desai-Hevia-Yin[20]은 ANSI X9.17 PRNG와 FIPS 186 PRNG에 대한 안전성을 규명하였다.

Ⅶ. 결 론

암호시스템에서 안전한 난수발생기를 사용하는 것은 안전성 관점에서 본질적인 요소이다. 대부분의 암호 알고리즘이나 암호 프로토콜은 난수발생기의 존재성을 가정하기 때문이다. 그러나 안전한 난수의 생성은 그렇게 쉽지만은 않은 작업이다.

본 논문에서는 하드웨어 기반의 RNG와 소프트웨어 기반의 PRNG를 사용되고 있는 현황과 최신 기술을 중심으로 조사해보았다. 또한, 출력 수열에 대한 통계적 난수성 평가 방법과 난수발생기를 하나의 알고리즘으로 보고 증명 가능 안전성 관점에서 분석하는 기술에 대하여 간략히 살펴보았다.

난수발생기는 블록이나 스트림, 공개키 암호 알고리즘, 그리고 각종 식별 및 인증 프로토콜들에 비해서 상대적으로 주목받지 못한 연구 분야였다. 그러나 최근에 NIST를 비롯한 표준화 그룹을 중심으로 그 중요성이 인식되어 활발한 연구가 진행되고 있다. 출력 수열에 대한 통계적 안전성 평가 방법은 이제 어느 정도 규격화된 모습을 갖추게 되었으나, 난수발생기에 대한 이론적 안전성 분석 기술은 향후에 많은 연구가 이루어져야 할 분야로 생각된다.

참 고 문 헌

〔1〕 FIPS 186-2, "Digital signature standard, Change notice 1", October 2001.
〔2〕 I. Goldberg, D. Wagner, "Randomness and the Netscape Browser", http://www.ddj.com/184409807, 2001.
〔3〕 T. Matthews, "Suggestions for random number generation in software", RSA Data Security Engineering Report, 1995.
〔4〕 P1363 Appendix E, "Cryptographic random numbers", 1995.
〔5〕 D. Eastlake, J. Schiller, S. Crocker, "Randomness requirements for security", IETF RFC 4086, June 2005.
〔6〕 E. Barker, J. Kelsey, "Recommendation for random number generation using deterministic random bit generators", NIST SP 800-90, December 2005.
〔7〕 R. Davis, "Hardware random number generators", Statistics Research Associates Limited, http://statsresearch.co.nz, October 2000.
〔8〕 B. Jun, P. Kocher, "The Intel Random Number Generator", Cryptography Research, Inc. White Paper, April 1999.
〔9〕 HotBits Hardware, http://www.fourmilab.ch/hotbits/hardware.html.
〔10〕 FIPS 186, "Digital Signature Standard",

NIST, 1994.
[11] ANSI X9.17, "American National Standard - Financial institution message authentication (wholesale)", 1986.
[12] RSA Data Security White Paper, "Using RSA BSAFE Crypto-C with the Intel Random Number Generator", http://www.rsasecurity.com/products/bsafe/intel/rsa_rng_tech.pdf.
[13] J. Kelsey, B Schneier, N. Ferguson, "Yarrow-160: Notes on the design and analysis of the Yarrow cryptographic pseudorandom number generator", Sixth Annual Workshop on Selected Areas in Cryptography, Springer Verlag, August 1999.
[14] Z. Gutterman, B. Pinkas, "Analysis of the Linux Random Number Generator", IEEE Symposium on Security and Privacy, May 2006.
[15] T. Dierks, C. Allen, "The TLS Protocol Version 1.0", IETF RFC 2246, January 1999.
[16] OpenSSH, http://www.openssh.com/
[17] S. W. Golomb, "Shift Register Sequences", Holden-Day, 1967.
[18] FIPS 140-1, "Security requirements for cryptographic modules", NIST, 1994.
[19] Special Publication 800-22, "A statistical test suite for random and pseudorandom number generators for cryptographic applications", NIST, May 2001.
[20] A. Desai, A. Hevia, Y. L. Yin, "A practice-oriented treatment of pseudo-random number generators", Eurocrypt 2002, LNCS 2332, 2002, pp. 368-383.

〈著 者 紹 介〉

강 주 성(Ju-Sung Kang)
정회원

1989년 고려대학교 수학과(학사)
1991년 고려대학교 일반대학원 수학과 (이학석사)
1996년 고려대학교 일반대학원 수학과 (이학박사)
1996년~1997년 과학재단 박사후연구원
1997년~2004년 한국전자통신연구원 선임연구원, 팀장
2001년~2002년 벨기에 루벤대학 COSIC 방문연구원
2004년~현재 국민대학교 수학과 부교수
〈관심분야〉 암호 알고리즘, 정보보호 프로토콜
e-mail : jskang@kookmin.ac.kr