creative commons

COMMONS DEED

저작자표시-비영리-동일조건변경허락 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.
- 이차적 저작물을 작성할 수 있습니다.

다음과 같은 조건을 따라야 합니다:

저작자표시. 귀하는 원저작자를 표시하여야 합니다.

비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.

동일조건변경허락. 귀하가 이 저작물을 개작, 변형 또는 가공했을 경우에는, 이 저작물과 동일한 이용허락조건하에서만 배포할 수 있습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 이용허락규약(Legal Code)을 이해하기 쉽게 요약한 것입니다.

Disclaimer

dCollection

[UCI]I804:24011-200000233084

2006年 2月
博士學位論文

의사난수에 대한 개선된 통계적 검증 방법

朝 鮮 大 學 校 大 學 院

數 學 科

林 廣 哲

의사난수에 대한 개선된 통계적 검증 방법

AN ADVANCED STATISTICAL TEST METHOD ON PSEUDO RANDOM NUMBERS

2006年 2月 日

朝 鮮 大 學 校 大 學 院

數 學 科

林 廣 哲

의사난수에 대한 개선된 통계적 검증 방법

指導敎授　鄭　潤　泰

이 論文을 理學博士學位申請 論文으로 提出함.

2005年　月　日

朝 鮮 大 學 校 大 學 院

數　學　科

林　廣　哲

林廣哲의 博士學位 論文을 認准함

委員長　　朝鮮大學校　教授　_____ 印

委　員　　朝鮮大學校　教授　_____ 印

委　員　　朝鮮大學校　教授　_____ 印

委　員　　朝鮮大學校　教授　_____ 印

委　員　　朝鮮大學校　教授　_____ 印

2005年　　月　　日

朝 鮮 大 學 校 大 學 院

목 차

목 차 ····· i
표 목 차 ····· iii
도 목 차 ····· iii
ABSTRACT ····· iv

제1장 서 론 ····· 1

제2장 표준암호알고리즘(라인돌) ····· 4
제1절 암호알고리즘을 이용한 의사난수 생성기 ····· 11
제2절 함수 생성기 ····· 12

제3장 이항분포와 정규분포 ····· 15
제1절 이항분포 ····· 15
제2절 정규분포 ····· 16
제3절 다항분포와 카이제곱 분포 ····· 17
제4절 적합도 검정과 카이제곱 분포 ····· 19

제4장 NIST 난수성 검정 ····· 21
제1절 프리퀀시 검정 ····· 21
제2절 런 검정 ····· 24

제3절 포커 검정 ····· 26
제4절 시리얼 검정 ····· 27

제5장 제안하는 난수성 검정 ····· 30

참고문헌 ····· 34

표 목 차

[표 1] 실난수와 의사난수의 비교 ····· 2
[표 2] 키 길이와 라운드 수의 관계 ····· 4

도 목 차

[그림 1] RIJNDAEL의 구조도 ····· 5
[그림 2] 흐름도 ····· 12
[그림 3] 입력데이타의 8비트 압축 ····· 13
[그림 4] 다항식 전개도 ····· 14

ABSTRACT

An Advanced Test Method on Pseudo Random Numbers

Rim, Kwang-Cheol
Advisor : Prof. Jung, Yoon-Tae Ph.D.
Department of Mathematics, Graduate School of Chosun University

Random numbers are used in many statistical procedures, in numerical mathematics, in physics, and also in number-theoretic applications to replace statistical observation or to automate the input of variable quantities.

In this paper we consider techniques for the statistical test and propose an advanced statistical test method. The advanced statistical test method is a kind of general serial test using χ^2 distribution.

This thesis is constituted as follows :

In chapter 2, we introduce the standard cipher algorithm, RIJNDAEL and the pseudo random number generator using cipher algorithm.

In chapter 3, we introduce the various results about polynomial distribution and normal distribution.

In chapter 4, we consider the various NIST random number tests, such as frequency test, run test, poker test, and serial test.

In chapter 5, we propose an advanced statistical test method , a kind of general serial test. We can reduce the number of operations of algorithm.

제1장 서론

암호학의 고전에는 주로 단순한 문자를 대입하고 이를 치환하여 정보를 은닉하였다. 이는 문장에 나타나는 문자들의 통계적 특성을 그대로 나타내기에 암호문의 통계적 특성을 분석하여 암호문을 해독할 수 있었다. 이후 복잡한 기계를 이용하여 암호문을 작성하였는데 이를 해독하기 위해서는 많은 양의 계산을 필요로 하였기에 그 시절에는 안전한 시스템이었다. 하지만 현대 정보전산능력이 뛰어난 시기에는 그도 또한 쉽게 해독할 수 있다. 이후 Shannon에 의해 복잡도가 높은 암호 알고리즘의 실현이 가능하게 되었고 현대 암호학의 기본 토대를 형성하게 되었다.[7][4]

현대암호는 크게 대칭키 암호시스템과 공개키 암호시스템으로 나눌 수가 있다. 1970년대 초 Shannon에 의해 주장된 혼돈(confusion)과 확산(diffusion)을 여러번 반복하면 강력한 암호 알고리즘을 구현할 수 있다는 이론에 의해 미국의 표준암호 알고리즘인 DES(Data Encryption Standard)가 IBM에 의해 제안되어 많은 기간 사용되었으며 이후 AES(Advanced Encryption Standard)로 발전하였다.[5][14][13][12]

일회용 암호(one-time pad)와 유사한 안전성을 보장하며 일반적인 통신망에도 적용할 수 있는 암호 알고리즘으로 스트림 암호 시스템이 있는데 이는 난수를 생성하여 평문과 1-1대칭으로 암복호화하는 방식이다. 이도 또한 엄밀한 의미로는 대칭키 암호 알고리즘으로 볼 수 있다.

대칭키 암호 알고리즘은 일단 키를 송신자와 수신자가 똑같이 나누어 가져야 한다는 불편이 있다. 이를 해소하기 위하여 암호화와 복호화과정에서 서로 다른 키를 사용하고 암호화키를 공개하여 키의 전송 및 비밀 보관 등이 필요 없게 만든 것이 공개키 암호 시스템이다. 이는 1976년 Diffe와 Hellman의 논문 “New Directions in Cryptography"에 발표가 되었다. 이도 또한 발전을 거쳐서 현재 RSA, ElGamal, 타원곡선암호, 땋임군 암호 등이 나와 있으나 계산양이 너무 많기 때문에 일반 평문에 대한 암호화는 힘들고 주로 키분배 알고리즘과 짧은 길이의 data에 대해 사용되고 있는 실정이다.[9][8][6][11]

의사난수 발생기는 거의 모든 암호학적 알고리즘에서 뼈대와 같이 사용되는 가장 중요한 암호학적 함수 중의 하나이다. 기존 의사난수 생성 알고리즘은 대수학적 이론에 기반을 둔 것과 하드웨어적 알고리즘에 기반을 둔 것으로 볼 수 가 있다.

의사난수 생성기는 스트림 암호의 원천을 이루고 또한 암호 프로토콜의 초기 벡터 또는 비밀 키, 전자 서명 및 전자 결제 시스템의 비밀 파라미터, 각종 키 관리/인증메커니즘에서의 세션키나 랜덤 첼린지(random challenge)의 생성 등에 사용된다.

난수에는 크게 실 난수(true randoms) 와 의사난수(pseudo randoms) 두 가지로 볼 수 있는데 그 차이점은 다음과 같다.

[표 1] 실난수와 의사난수의 비교

실난수(*true random numbers*)	의사난수(*pseudo-random numbers*)
l 비결정적이다. l 예측 불가능한 어떤 물리적인 소스로부터 획득 l (예) 반도체, 방사선 붕괴 등으로부터 전자소음 혹은 열소음, 등	l 컴퓨터는 논리적이고 결정적이므로, 실난수를 산출하지 못함. l S/W에 기반한 RNG는 최상의 경우에, 의사난수를 생성 가능. l PRNG : 길이가 짧은 랜덤 비트열(seed)을 길이가 긴 랜덤에 근접한 비트열로 출력하는 알고리즘

난수성을 만족하는 좋은 난수의 특징은 패턴과 역상관관계를 알 수 없어야 하고 긴 주기를 가져야 하며 1과 0의 평균생성횟수가 동일하여야 한다. 그리고 "01010101"과 같은 형태가 길지 않아야 하고 간단한 알고리즘으로 설계와 구현이 쉬워야 한다. 마지막으로 난수의 특징은 이미 나온 출력으로부터 그 전의 값을 유추할 수 없어야 한다.

이러한 난수성을 만족하고 구현상의 용이함을 위하여 암호학적 알고리즘을 이용한 의사난수 생성기를 사용한다. 암호학적 알고리즘을 이용하면 먼저 암호학적 키의 안전도에 의하여 난수를 선택하는 기준을 정할 수 있고 암호학적 알고리즘(encryption algorithm)이나 보안 프로토콜(security protocols)에는 관심이 많으나

실상 키를 담당하는 의사난수 생성에는 소홀할 수 있는 단점을 가지고 있다. 하지만 현대의 공격자나 해커들은 복잡한 알고리즘보다 랜(ran)의 공격에 주력하기 때문에 암호학적 알고리즘을 통한 의사난수 생성기는 기본적인 안전에 대한 믿음을 제공한다고 볼 수 있다.

이로써 생성되는 의사난수들은 실 난수와 차별성을 가지기 때문에 엄밀히 말하면 난수라고 보기는 어렵다. 우선 의사난수 생성기에 의하여 생성되는 비트열들은 우선 유한 차원의 주기를 갖는다. 하지만 이 유한 차원의 주기를 무한히 확장하면 이는 실난수와의 차별성을 어느 정도 완화한다고 볼 수가 있다. 이에 이러한 의사난수 생성기에 의해 생성되는 난수열들을 실난수와 같은 성질을 가짐을 보이기 위하여 실난수의 특징들을 적용하여 테스트할 필요성을 갖는다.

본 논문에서는 차세대 암호표준인 라인돌(RIJNDAEL)에 대하여 살펴보고 이를 이용한 의사난수 생성기의 기본 구조와 트리구조를 이용한 의사난수 함수열과 이를 통한 의사난수 비트열의 생성에 대하여 알아본다. 이러한 의사난수 생성기에 의하여 생성되는 의사난수비트열에 대하여 기존에 알려져있는 의사난수열에 대한 통계학적인 테스트들인 프리퀀시 검정, 런 검정, 시리얼 검정 등에 대하여 살펴보고 이를 개선한 새로운 의사난수 통계적 테스트를 제안하고 이에 대한 효율성을 계산해 보았다.

제2장 표준암호알고리즘(라인돌)

라인돌은 벨기에의 수학자 Daemen과 Rijmen에 의하여 개발되어 2000년 10월 AES(Advanced Encryption Standard)로 최종 선정 되었다.[3][1] 블록의 길이와 key 길이가 각각 128, 192, 256의 가변적인 성질을 가지고 있으며 블록과 키 길이에 따라 라운드의 수가 결정된다. 라인돌에서는 상태(State)라 불리는 중간 값에 여러 가지 변환을 적용한다. 상태는 바이트의 직사각형 배열로 나타낼 수 있다. 암호 키도 마찬가지 형태로 나타낸다.

[표 2] 키 길이와 라운드 수의 관계

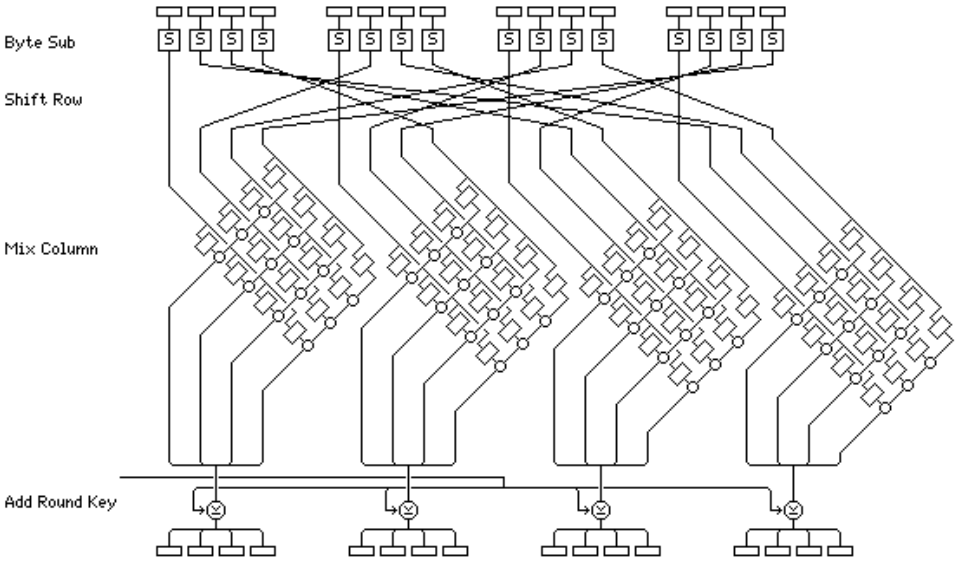
	키 길이(N_k)	블록 크기(N_b)	라운드 수 (N_r)
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

배열의 행수는 4이며 열 수는 데이터의 경우 N_b, 키의 경우 N_k로 나타낸다. 라운드 수의 변화는 표 1과 같다.

라인돌은 첫 라운드 키 더하기, N_{r-1}번의 라운드, 최종 라운드로 구성된다. N_{r-1}번의 라운드 중 한 라운드와 최종 라운드의 알고리즘은 다음과 같다.

```
Round (State, RoundKey)
{
   ByteSub (State) ;
   ShiftRow (State) ;
   MixColumn (State) ;
   AddRoundKey (State, RoundKey) ;
}
```

```
FinalRound (State, RoundKey)
{
   ByteSub (State) ;
   ShiftRow (State) ;
   AddRoundKey (State, RoundKey) ;
}
```



[그림 1] 라인돌의 구조도

ByteSub 변환은 상태의 각 바이트에 독립적으로 비선형 바이트 치환을 행하는 것이다. ShiftRow 변환은 상태의 각 행을 정해진 수만큼 쉬프트 하는 것이다. MixColumn 변환은 상태의 각 열을 유한체 GF(2^8)에서의 다항식으로 간주하여 다음으로 주어지는 c(x)를 곱한 후 x^4+1에 대해 모듈러 연산을 한 결과이다.

$$c(x) = 3x^3 + x^2 + x + 2$$

AddRoundKey 변환에서는 XOR를 사용하여 라운드 키를 상태에 적용한다.
각 라운드 키는 키 스케쥴에 의해 암호화키로부터 만들어진다. 키 스케줄은 키 확장과 라운드 키 선택으로 이루어진다. 확장된 키는 4바이트 워드의 선형 배열로 생각할 수 있으며 W[$N_b*(N_{r+1})$]로 나타낼 수 있다. N_k가 6이하인 경우의 키 확장은 다음과 같은 알고리즘을 따른다.

```
KeyExpansion(byte Key[4*Nk], word W[Nb*Nr+1)])
{
  for(i=0; i<Nk; i++)
    W[i] = (Key[4*i], Key[4*i+1], Key[4*i+2], Key[4*i+3]);
  for(i=Nk; i<Nb*(Nr+1); i++)
  {
    temp = W[i-1];
    if(i%Nk == 0)
      temp = SubByte(RotByte(temp)) ^ Rcon[i/Nk];
    W[i] = W[i-Nk]^temp;
  }
}
```

라인돌을 128비트 시스템으로 암호화와 복호화를 시행하면 다음과 같다.

암호화 (라운드 수 *r = 0 to 10*):	복호화 (라운드 수 *r = 0 to 10*):
input: 평문 입력	iinput: 복호화문 입력(inverse cipher input)
start: r 라운드의 시작상태	istart: r 라운드 시작 상태
s_box: SubBytes()	is_box: InvSubBytes()
s_row: ShiftRows()	is_row: InvShiftRows()
m_col: MixColumns()	ik_sch: 키 스케줄후 키값
k_sch: 키 스케줄후 키값	ik_add: AddRoundKey()
output: 출력	ioutput: 출력

AES-128 (Nk=4, Nr=10)

평문(PLAINTEXT): 00112233445566778899aabbccddeeff

KEY: 000102030405060708090a0b0c0d0e0f

CIPHER (ENCRYPT) : 36

round[0].input 00112233445566778899aabbccddeeff

round[0].k_sch 000102030405060708090a0b0c0d0e0f

round[1].start 00102030405060708090a0b0c0d0e0f0

round[1].s_box 63cab7040953d051cd60e0e7ba70e18c

round[1].s_row 6353e08c0960e104cd70b751bacad0e7

round[1].m_col 5f72641557f5bc92f7be3b291db9f91a

round[1].k_sch d6aa74fdd2af72fadaa678f1d6ab76fe

round[2].start 89d810e8855ace682d1843d8cb128fe4

round[2].s_box a761ca9b97be8b45d8ad1a611fc97369

round[2].s_row a7be1a6997ad739bd8c9ca451f618b61

round[2].m_col ff87968431d86a51645151fa773ad009

round[2].k_sch b692cf0b643dbdf1be9bc5006830b3fe

round[3].start 4915598f55e5d7a0daca94fa1f0a63f7

round[3].s_box 3b59cb73fcd90ee05774222dc067fb68

round[3].s_row 3bd92268fc74fb735767cbe0c0590e2d

round[3].m_col 4c9c1e66f771f0762c3f868e534df256

round[3].k_sch b6ff744ed2c2c9bf6c590cbf0469bf41

round[4].start fa636a2825b339c940668a3157244d17

round[4].s_box 2dfb02343f6d12dd09337ec75b36e3f0

round[4].s_row 2d6d7ef03f33e334093602dd5bfb12c7

round[4].m_col 6385b79ffc538df997be478e7547d691

round[4].k_sch 47f7f7bc95353e03f96c32bcfd058dfd

round[5].start 247240236966b3fa6ed2753288425b6c

round[5].s_box 36400926f9336d2d9fb59d23c42c3950

round[5].s_row 36339d50f9b539269f2c092dc4406d23

round[5].m_col f4bcd45432e554d075f1d6c51dd03b3c

round[5].k_sch 3caaa3e8a99f9deb50f3af57adf622aa

round[6].start c81677bc9b7ac93b25027992b0261996

round[6].s_box e847f56514dadde23f77b64fe7f7d490

round[6].s_row e8dab6901477d4653ff7f5e2e747dd4f

round[6].m_col 9816ee7400f87f556b2c049c8e5ad036

round[6].k_sch 5e390f7df7a69296a7553dc10aa31f6b

round[7].start c62fe109f75eedc3cc79395d84f9cf5d

round[7].s_box b415f8016858552e4bb6124c5f998a4c

round[7].s_row b458124c68b68a014b99f82e5f15554c

round[7].m_col c57e1c159a9bd286f05f4be098c63439

round[7].k_sch 14f9701ae35fe28c440adf4d4ea9c026

round[8].start d1876c0f79c4300ab45594add66ff41f

round[8].s_box 3e175076b61c04678dfc2295f6a8bfc0

round[8].s_row 3e1c22c0b6fcbf768da85067f6170495

round[8].m_col baa03de7a1f9b56ed5512cba5f414d23

round[8].k_sch 47438735a41c65b9e016baf4aebf7ad2

round[9].start fde3bad205e5d0d73547964ef1fe37f1

round[9].s_box 5411f4b56bd9700e96a0902fa1bb9aa1

round[9].s_row 54d990a16ba09ab596bbf40ea111702f

round[9].m_col e9f74eec023020f61bf2ccf2353c21c7

round[9].k_sch 549932d1f08557681093ed9cbe2c974e

round[10].start bd6e7c3df2b5779e0b61216e8b10b689

round[10].s_box 7a9f102789d5f50b2beffd9f3dca4ea7

round[10].s_row 7ad5fda789ef4e272bca100b3d9ff59f

round[10].k_sch 13111d7fe3944a17f307a78b4d2b30c5

round[10].output 69c4e0d86a7b0430d8cdb78070b4c55a

INVERSE CIPHER (DECRYPT):

round[0].iinput 69c4e0d86a7b0430d8cdb78070b4c55a

round[0].ik_sch 13111d7fe3944a17f307a78b4d2b30c5

round[1].istart 7ad5fda789ef4e272bca100b3d9ff59f37

round[1].is_row 7a9f102789d5f50b2beffd9f3dca4ea7

round[1].is_box bd6e7c3df2b5779e0b61216e8b10b689

round[1].ik_sch 549932d1f08557681093ed9cbe2c974e

round[1].ik_add e9f74eec023020f61bf2ccf2353c21c7

round[2].istart 54d990a16ba09ab596bbf40ea111702f

round[2].is_row 5411f4b56bd9700e96a0902fa1bb9aa1

round[2].is_box fde3bad205e5d0d73547964ef1fe37f1

round[2].ik_sch 47438735a41c65b9e016baf4aebf7ad2

round[2].ik_add baa03de7a1f9b56ed5512cba5f414d23

round[3].istart 3e1c22c0b6fcbf768da85067f6170495

round[3].is_row 3e175076b61c04678dfc2295f6a8bfc0

round[3].is_box d1876c0f79c4300ab45594add66ff41f

round[3].ik_sch 14f9701ae35fe28c440adf4d4ea9c026

round[3].ik_add c57e1c159a9bd286f05f4be098c63439

round[4].istart b458124c68b68a014b99f82e5f15554c

round[4].is_row b415f8016858552e4bb6124c5f998a4c

round[4].is_box c62fe109f75eedc3cc79395d84f9cf5d

round[4].ik_sch 5e390f7df7a69296a7553dc10aa31f6b

round[4].ik_add 9816ee7400f87f556b2c049c8e5ad036

round[5].istart e8dab6901477d4653ff7f5e2e747dd4f

round[5].is_row e847f56514dadde23f77b64fe7f7d490

round[5].is_box c81677bc9b7ac93b25027992b0261996

round[5].ik_sch 3caaa3e8a99f9deb50f3af57adf622aa

round[5].ik_add f4bcd45432e554d075f1d6c51dd03b3c

round[6].istart 36339d50f9b539269f2c092dc4406d23

round[6].is_row 36400926f9336d2d9fb59d23c42c3950

round[6].is_box 247240236966b3fa6ed2753288425b6c

round[6].ik_sch 47f7f7bc95353e03f96c32bcfd058dfd

round[6].ik_add 6385b79ffc538df997be478e7547d691

round[7].istart 2d6d7ef03f33e334093602dd5bfb12c7

round[7].is_row 2dfb02343f6d12dd09337ec75b36e3f0

round[7].is_box fa636a2825b339c940668a3157244d17

round[7].ik_sch b6ff744ed2c2c9bf6c590cbf0469bf41

round[7].ik_add 4c9c1e66f771f0762c3f868e534df256

round[8].istart 3bd92268fc74fb735767cbe0c0590e2d

round[8].is_row 3b59cb73fcd90ee05774222dc067fb68

round[8].is_box 4915598f55e5d7a0daca94fa1f0a63f7

round[8].ik_sch b692cf0b643dbdf1be9bc5006830b3fe

round[8].ik_add ff87968431d86a51645151fa773ad009

round[9].istart a7be1a6997ad739bd8c9ca451f618b61

round[9].is_row a761ca9b97be8b45d8ad1a611fc97369

round[9].is_box 89d810e8855ace682d1843d8cb128fe4

round[9].ik_sch d6aa74fdd2af72fadaa678f1d6ab76fe

round[9].ik_add 5f72641557f5bc92f7be3b291db9f91a

round[10].istart 6353e08c0960e104cd70b751bacad0e7

round[10].is_row 63cab7040953d051cd60e0e7ba70e18c

round[10].is_box 00102030405060708090a0b0c0d0e0f0

round[10].ik_sch 000102030405060708090a0b0c0d0e0f

round[10].ioutput 00112233445566778899aabbccddeeff

제1절 암호알고리즘을 이용한 의사난수 생성기

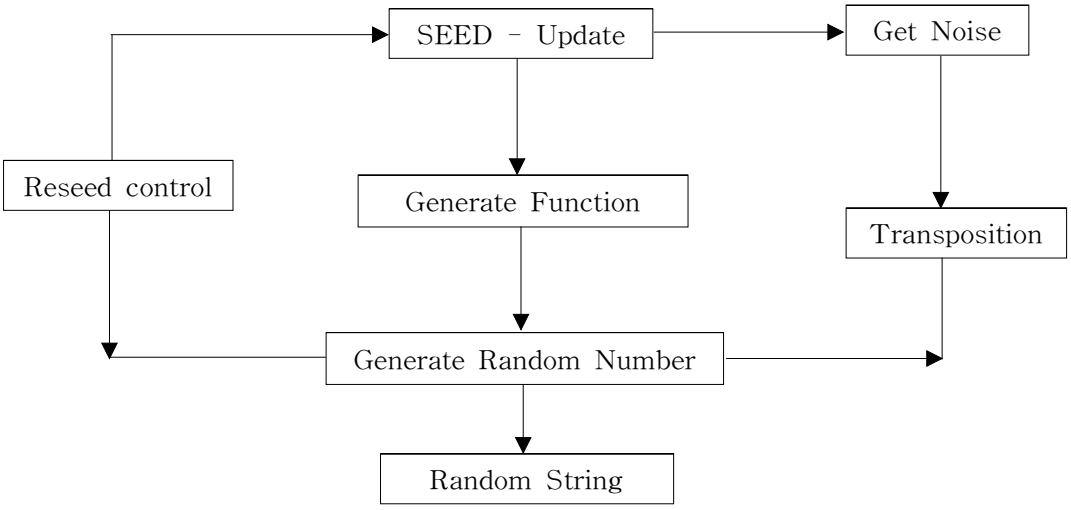
좋은 의사난수 생성기는 입력과 출력을 통한 공격을 막기 위한 방법으로 중간 상태변수를 알 수 없게 하여야 한다. 이를 위해서는 초기값과 상태 값들을 일정 출력과 일정 시간이 흐른 후에는 갱신과정을 통하여 새롭게 입력되고 변화하여야한다. 또한 의사난수 생성기에 의한 출력물들이 기존에 알려진 공격에 강해야 한다.

제안된 의사난수 생성기는 기존의 알려진(검증된) 암호알고리즘을 통하여 입력값의 변화와 각각 함수들의 조합으로 출력이 결정되므로 알려진 공격들에 대하여 안전도 면에서 입증이 되었고 또한 매 일정 출력이 실행된 후에는 새로운 입력 값을 받아서 난수생성이 진행되므로 중간 상태변수의 노출 염려가 보다 줄어지는 효과를 볼 수 있다.

제안된 의사난수 생성기는 라인돌의 입력과 출력에 맞추어서 진행하였는데 먼저 초기값 재설정(seed-update)에서 실난수(true random number) 128bit를 입력으로 받아서 이를 압축알고리즘을 통한 데이터 압축으로 함수생성기에 보내진다. 초기값 재설정 과정은 출력물 10000bit 이후 재 진행되고 또한 출력 카운터 100회 마다 재 생성되도록 설계하여 공격자들의 중간 값에 대한 변화와 이를 통한 공격을 최소화 하였다. 버퍼메모리를 사용하지 않고 중간 데이터 값들이 생성 후 기록되지 않게 설계하였다. 재설정 과정을 거치므로 인해서 상태변수와 중간 데이터 노출에 대한 안전도를 향상시켰으며 일정 카운터에 대한 새로운 입력과정을 두어 입력 데이터의 주기적인 변화를 추구하였다.

잡음입력(get noise)와 변환(transposition)에 있어서는 들어온 입력 초기값(seed)으로부터 128bit를 취하여 해쉬함수를 통한 비트(bit)변환을 거친 후 생성된 난수 생성함수의 키(key)로써 작용한다. 이도 또한 일정 카운터와 일정 출력 후에는 새로운 잡음원을 취하여 새로운 키를 생성해준다.

흐름도는 다음 [그림3] 과 같다.



[그림2] 흐름도

제2절 함수 생성기

제안하는 함수 생성기는 먼저 유한체 GF(2^8)상에서의 모듈연산을 통하여 생성된 함수를 라인돌과 같은 4차항의 기약다항식으로 나누어 그 나머지를 연산자로 이용하는 구조를 가진다.

초기값재설정(seedupdate) 과정에서 128비트의 실난수를 입력받아서 16비트 8블럭으로 분할한 후 각 16비트 블록에서 8비트를 생성해 낸다. 첫 번째 블록의 1비트 자리와 두 번째 블록의 5비트자리, 세 번째 비트의 2비트자리 네 번째 블록의 6비트자리에서 4비트를 생성해 낸다. 또한 다섯 번째 블록의 3비트 자리, 여섯 번째 블록의 7비트자리와 일곱 번째 블록의 4비트 자리, 여덟 번째 블록의 8비트 자리에서 4비트를 생성한다. 전체적인 비트 압축의 구조도는 아래와 같다.

16비트		16비트		16비트		16비트		16비트		16비트		16비트		16비트	
1			5	2			6	3			7	4			8

[그림 3] 입력데이터의 8비트 압축

예를 들면 128비트 d5014e43 60ef2bac 96211b40 72ba3224(16진수표현) 에서 8비트 10110010을 얻는다. 생성된 8비트는 GF(2^8)상의 다항식으로 전환하여 트리함수를 따라 새로운 다항식으로 변환한다. 이 후 이를 A^4+1로 나눈 나머지를 의사난수 생성함수로 사용한다. 그 사용은 앞 계수는 쉬프트연산 방향과 진행비트를 나타내고 각 항들은 배타적 논리합으로 이루어진다.

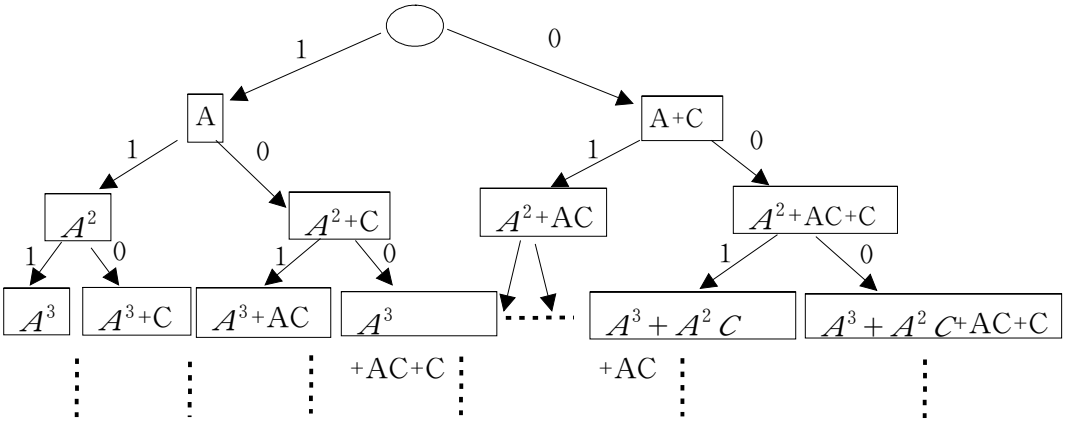
이는 기존 알려진 공격법에 대한 선형적인 합성과 암호알고리즘들의 비선형적 결합의 조합으로 비도를 높이고 중간공격을 차단할 수 있다. 또한 초기 입력 값으로부터 해쉬함수를 통한 입력키 값을 산출해 냄으로 인해서 입력 데이터와의 차별성을 두었다.

초기 128비트의 입력에서 압축 후 8비트 10110011을 얻었다. 이를 상수 C가 3인 값으로 정하여 트리함수에 적용하면 $A^8+3A^6+3A^3+A^2$을 얻는다. 이를 기약다항식 A^4+1로 나눈 나머지는 $3A^3-2A^2-1$ 이다.

이는 라인돌을 3회 시행 후 오른쪽 3비트 쉬프트를 취한 후 다시 2회 라인돌을 구현하고 좌측 2비트 쉬프트 연산을 수행한 후 상수 1을 뺀다.

이를 통한 의사난수를 생성한 후 NIST의 통계 검정 패키지를 통한 난수성을 검정해 보았다. 기존의 라인돌이 이미 난수성을 가진 것으로 판명이 나 있으므로 난수성에 대한 보증은 장담하리라 본다.

이 시스템은 알려진 공격에 대한 난수생성에 의한 탬플릿들이 공격에 강하고 대수학적 특징을 이용한 것보다 구현과 비도면에서 더 강한 알고리즘임이 증명이 돼 있으므로 고사양의 암호 알고리즘 구축과 보안 시스템에서의 사용이 좋을 것으로 보인다.



[그림 4] 다항식 전개도

향후 기존 암호알고리즘의 단위 라운드별로의 설계를 구현하여 속도와 안전도를 동시에 만족하는 시스템 개발도 의미있는 일이라 사료된다.[2][10]

제3장 이항 분포와 정규분포

제1절 이항분포

어느 실험이 오직 두 가지의 가능한 결과(outcomes)만을 가질 때 이를 베르누이 시행(Bernoulli trial)이라 한다. 시행의 결과는 일반적으로 "성공(success)"과 "실패(failure)"로 나타낸다. 성공률이 p인 베르누이 시행을 n번 독립적으로 수행했을 때 확률변수 X를 성공횟수라고 하자. 이때 X의 확률 분포를 시행횟수 n과 성공률 p를 갖는 이항분포(binomial distribution)라 하고 기호로는 $B(n,p)$로 나타낸다. 즉, $k=0,1,\cdots,n$에 대하여 성공이 k번 발생할 확률은

$$P(X=k)=\binom{n}{k}p^k(1-p)^{n-k},$$

$$,\ \binom{n}{k}=\frac{n!}{k!\,(n-k)!}$$

이 된다. 간단한 계산에 의하여 X의 기대값과 분산을 구하면 다음과 같다.

$$E[X]=np,\quad Var[X]=np(1-p).$$

이항분포 $B(n,p)$를 따르는 확률변수 X에 대하여 n이 충분히 크고 p가 0 또는 1에 가깝지 않을 때 확률변수

$$\frac{X-np}{\sqrt{np(1-p)}}$$

의 분포는 근사적으로 표준 정규분포 $N(0,1)$을 따른다는 사실이 잘 알려져 있다.

제2절 정규분포

몸무게나 키와 같이 자연현상에서 측정할 수 있는 연속확률변수는 대개 종모양의 분포를 가진다. 즉 이들의 분포는 평균근처에서 가장 높은 확률을 보이고, 평균에서 멀어질수록 대칭적으로 확률이 작아진다. 가우스(Carl Gauss)는 종 모양을 보이는 분포의 확률밀도함수를 고안하였는데, 이러한 종 모양의 분포를 가우스분포(Gaussian distribution) 또는 정규분포(normal distribution)라고 한다. 자연현상에서 관찰되는 관측값들의 분포는 대개 종 모양을 띠고, 따라서 정규분포를 따른다고 가정하는 것이 일반적이다.

확률밀도함수: $f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/2\sigma^2}, \quad -\infty < x < +\infty$

식이 위와 같을 때 X가 평균이 μ이고 분산이 σ^2인 정규분포(normal distribution)을 갖는다 라고 한다. (여기서 x는 확률변수 X의 값이고, μ는 분포의 산술평균이다 그리고 σ는 분포의 표준편차이다.)

위 정규분포의 확률밀도함수에서 평균이 0($\mu = 0$), 분산이 1($\sigma^2 = 1$)인 정규분포를 표준 정규분포 라고 한다. (즉, 모든 정규분포를 평균 μ=0, 표준편차 σ=1이 되도록 표준화한 것)

그래서 X는 "표준정규분포(standard normal distribution)를 갖는다" 라고 한다. 어떤 관측치 x가 그 분포의 평균으로부터 표준편차의 몇 배 정도나 떨어져 있는가를 다음과 같이 표준화된 정규분포확률변수 Z로 나타내기 때문에 표준정규분포를 따르는 확률 변수는 보통 Z로 표시한다. 수식으로 간단히 표시하면 $Z \sim N(0,1)$이다.

확률밀도함수: $f(z) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}z^2}, \quad -\infty < z < \infty$

제3절 다항분포와 카이제곱분포

실험의 결과가 둘 중의 하나로 나타나는 베르누이시행을 일반화하여 이제 나타날 수 있는 결과가 여러 개 중의 하나인 시행을 살펴보자. 이렇게 나타날 수 있는 결과가 여러 개인 시행을 다항시행(multinomial trial)이라고 부른다. 한 번의 다항시행에서 k개의 서로 배반일 사건이 발생한다고 하고, 이들 중 i번째 사건이 일어날 확률을 p_i라고 하자 $(1 \le i \le k)$. n번의 서로 독립인 다항시행을 행하여 i번째 사건이 일어날 도수(frequency)를 X_i $(i = 1, \cdots, k)$라고 하면

$$P(X_1 = n_1, \cdots, X_k = n_k) = \frac{n!}{n_1! \cdots n_k!} p_1^{n_1} p_2^{n_2} \cdots p_k^{n_k}$$

임을 알 수 있다. 여기에서 $n_1 + \cdots + n_k = n$이다. 이때, $(X_1, \cdots, X_k)$는 모수(parameter) $(n, p_1, \cdots, p_k)$를 갖는 다항 분포(multinomial distribution)를 따른다고 하며 기호로는

$$(X_1, \cdots, X_k) \sim m(n, p_1, \cdots, p_k)$$

로 나타낸다.

간단한 계산에 의해서 $X_1, \cdots, X_k$의 기댓값, 분산, 공분산(covariance)을 구하면 다음과 같다. $1 \le i \le k$인 각각의 i에 대해서

$$E[X_i] = np_i, \quad Var[X_i] = np_i(1 - p_i)$$

이고, $1 \le i, j \le k$와 $i \ne j$를 만족하는 i, j에 대해서

$$Cov(X_i, X_j) = -np_ip_j$$

가 된다.

이항분포가 근사적으로 정규분포에 수렴함을 알기에 이와 비슷하게 다항분포는 카이제곱분포와 연관되어 있다는 사실에 대해 알아보자. $(X_1, X_2) \sim m(n, p_1, p_2)$라고 하면, $X_2 = n - X_1$, $p_2 = 1 - p_1$이고 $X_1 \sim B(n, p_1)$이다. 그러므로

$$\begin{aligned}\sum_{i=1}^{2} \frac{(X_i - np_i)^2}{np_i} &= \frac{(X_1 - np_1)^2}{np_1} + \frac{(n - X_1 - n(1-p_1))^2}{np_2} \\ &= \frac{(X_1 - np_1)^2}{np_1} + \frac{(-X_1 + np_1)^2}{np_2} \\ &= \frac{p_2(X_1 - np_1)^2 + p_1(X_1 - np_1)^2}{np_1p_2} \\ &= \left(\frac{X_1 - np_1}{\sqrt{np_1(1-p_1)}} \right)\end{aligned}$$

이 성립한다. 왜냐하면 $p_1 + p_2 = 1$ 그런데 n이 충분히 클때

$\frac{X_1 - np_1}{\sqrt{np_1(1-p_1)}} \sim N(0,1)$이므로 카이제곱 분포의 정의와 식에 의해

$$\sum_{i=1}^{2} \frac{(X_i - np_i)^2}{np_i} = \left(\frac{X_1 - np_1}{\sqrt{np_1(1-p_1)}} \right)^2 \sim \chi^2(1)$$

임을 알 수 있다. 여기에서 $\chi^2(r)$은 자유도(degree of freedom)가 r인 카이 제곱 분포를 의미한다.

위의 논리를 귀납적으로 다항분포 $m(n, p_1, \cdots, p_k)$를 따르는 $(X_1, \cdots, X_k)$에 적용하면,

$$X_k = n - (X_1 + \cdots + X_{k-1}), \ p_k = 1 - (p_1 + \cdots + p_{k-1})$$

이고, $1 \le i \le k-1$인 각 i에 대해서 $X_i \sim B(n, p_i)$ 라는 사실에 의해서 다음 정리를 얻을 수 있다.

<정리> 다항분포 $(X_1, \cdots, X_k) \sim m(n, p_1, \cdots, p_k)$ 일 때, n이 충분히 크면 근사적으로 $\sum_{i=1}^{k} \frac{(X_i - np_i)^2}{np_i} \sim \chi^2(k-1)$을 만족한다.

정리의 근사분포는 실용적으로 $np_i > 5$ $(i = 1, \cdots, k)$ 일 때에 적용하는 것이 좋은 것으로 알려져 있다.

제4절 적합도 검정과 카이제곱분포

어떤 관측 결과가 특정한 분포로부터 관측된 값인지를 평가하는 검정법을 적합도 검정(goodness-of-fit test)이라 일컫는다. 일반적으로 관측도수에 대한 적합도 검정으로 카이제곱검정이 이용되는데 그 이론적 배경을 설명하기로 하자.

n개의 실험 대상을 택하여 서로 배반인 k개의 범주(category)에 속하는 도수 $X_1, \cdots, X_k$를 관측 한다고 하자. 각 실험 대상이 i번째 범주에 속할 확률을 $p_i (i = 1, \cdots, k)$라고 하면 도수를 나타내는 확률 변수 $(X_1, \cdots, X_k)$는 다항 분포 $m(n, p_1, \cdots, p_k)$를 따른다.

이제 주어진 값 $p_{10}, \cdots, p_{k0}$ $(p_{10} + \cdots + p_{k0} = 1,\ p_{i0} \ge 0)$에 대하여 귀무가설(null hypothesis) H_0와 대립 가설(alternative hypothesis) H_1이 각각

$$H_0 : p_1 = p_{10}, \cdots, p_k = p_{k0},\ vs\ \ H_1 : H_0\text{가 아니다.}$$

로 주어지는 경우의 적합도 검정을 생각해 보자. 귀무 가설 H_0가 사실이라면, i번째 범주의 기대 도수(expected frequency)는

$$E[X_i \mid H_0] = np_{i0}, \quad i=1,\cdots,k$$

로 주어진다. 따라서 실제 관측도수(observed frequency) $X_i = x_i$가 기대도수 np_{i0}와 차이가 나면 H_0가 옳지 않음을 뜻한다. 이러한 기대 도수와 관측 도수의 차이에 대한 측도로서

$$\sum_{i=1}^{k} \frac{(x_i - np_{i0})^2}{np_{i0}}$$

이 이용된다. 이것이 사용되는 이유는 다항분포의 성질에서 H_0가 사실이라면 근사적으로

$$\sum_{i=1}^{k} \frac{(X_i - np_{i0})^2}{np_{i0}} \sim \chi^2(k-1)$$

을 만족하기 때문이다.

결과적으로 관측도수의 적합도 검정에서 검정통계량(statistics)을

$$\sum_{i=1}^{k} \frac{(X_i - np_{i0})^2}{np_{i0}}$$

으로 사용한다면, 주어진 유의수준과 자유도가 $k-1$인 카이제곱 분포표에 의해 기각역(critical region)을 결정할 수 있다.

제4장 NIST 난수성 검정

NIST에서 제공하는 의사난수 테스트에는 16가지가 있다. 여기서는 앞부분의 중요한 프리퀀시 검정, 런 검정, 포커 검정, 시리얼 검정에 대하여 살펴보기로 한다.

제1절 프리퀀시 검정

이진 수열 (s_t)의 N개의 항 $s_0,\ s_1, \cdots\ s_{t+2},\ \cdots, s_{N-1}$을 처음부터 차례로 m개씩 나누면, 다음과 같이 m차원 벡터공간 $GF(2)^n$에 속하는 $n=[\frac{N}{m}]$개의 벡터가 생긴다.

$$
\begin{array}{ll}
 & (s_0,\ s_1, \cdots,\ s_{t+2},\ \cdots, s_{m-1}) \\
* & (s_m,\ s_{m+1},\ s_{m+2},\ \cdots, s_{2m-1}) \\
 & \cdots\cdots\cdots\cdots\cdots\cdots \\
 & (s_{(n-1)m},\ s_{(n-1)m+1},\ \cdots, s_{nm})
\end{array}
$$

이제 체 $GF(2) = \{0,1\}$ 위의 m 차원 벡터공간

$$GF(2)^n = \{(x_1, x_2, \cdots, x_m) \mid x_1, x_2, \cdots, x_m \in GF(2)\}$$

에 속하는 2^m개의 벡터를 적당히 번호를 정하여, (*)에 있는 벡터중에서

$(0,0,\cdots,0)$인 것의 개수를 $n(0)$

$(0,0,\cdots,1)$인 것의 개수를 $n(1)$

……

$(1,1,\cdots,1,1)$인 것의 개수를 $n(2^m-1)$

라고 하자. 이 때, N개의 항으로 이루어진 유한 이진 수열

$$s_0, s_1, \cdots, s_{t+2}, \cdots, s_{N-1}$$

이 완전한 random 수열인 경우에는 $GF(2)^n$에 속해 있는 각 벡터가 (*) 에 있는 n개의 벡터 중의 하나로 나타날 확률은 $\frac{n}{2^m}$이다.

따라서 통계량

$$T=\sum_{i=0}^{2^m-1}\frac{\left\{n(i)-\frac{n}{2^m}\right\}^2}{\frac{n}{2^m}}=\frac{2^m}{n}\sum_{i=0}^{2^m-1}n(i)^2-n$$

은 근사적으로 자유도가 2^m-1인 χ^2분포를 따른다.

이제 $0<\alpha<1$ 인 α에 대하여

$$P(\chi^2\geq x_\alpha)=\alpha$$

인 실수 x_α 를 χ^2 분포표에서 구하면, 유의수준 α에 대하여 기각역은 $T>x_\alpha$ 이다.

예를 들어, $\alpha=0.01$에 대하여

$$P(\chi^2\geq x_\alpha)=0.01$$

일 때, T 의 값이 α 보다 크면 주어진 수열은 랜덤(random) 하지 않다고 말 할 수 있고 T 의 값이 α 보다 작으면 이 검정법으로는 랜덤(random) 하지 않다고 판정할 수는 없다.

이와 같은 검정법을 프리퀀시 엠검정(m-test) 이라고 한다.

일반적으로, 모든 m에 대하여 일일이 이와 같은 검정법을 시행할 수는 없으므로 경우에 따라 m 의 값을 적절히 택한다.

특히, m=1 인 경우에 이 검정법을 흔히 프리퀀시 검정(frequency-test) 이라고 한다. 이 경우에 식은 다음과 같다.

$$T = \frac{2}{N}\sum_{i=1}^{1}\left(n(i) - \frac{N}{2}\right)^2$$

여기서 $n(0)$ 과 $n(1)$은 각각 $s_0, s_1, s_2, \cdots, s_{N-1}$ 중에서 0인 것의 개수와 1인 것의 개수를 뜻한다. 이제 $n_0 = n(0)$, $n_1 = n(1)$ 이라고 놓으면,

$$N = n = n(0) + n(1) = n_0 + n_1$$

이므로, 위의 식은 다음과 같이 변형된다.

$$T = \frac{(n_0 - n_1)^2}{N}$$

그리고, 이 경우에 통계량

$$Z = \frac{1}{\sqrt{\frac{n}{4}}}\left\{n_1 - \frac{n}{2}\right\}$$

는 근사적으로 표준정규분포 $\mathcal{N}(0,1)$ 을 따른다.

한편, χ^2분포에서 유의수준 5%의 한계값은 3.84 이다. 따라서 T의 값이 3.84 보다 큰 경우에, 프리퀀시 검정에 대하여 유의수준 5%로 이 이진수열은 난수성이 없다고 판단되어 기각한다. 이에 대한 판단으로 유의확률 피벨류(p-value)를 사용하는데 피벨류 0.01 이상에 대하여 유의수준을 결정한다.

예를 보면 다음과 같다.

입력 수열 e =1100100100001111110110101010001000100001011010001100000100011010011000100110001100110001010001011100

n = 100

Sn = -16

sobs = 1.6

p-value = 0.109599

결과 : p-value > 0.01 이므로 이 수열은 랜덤하다고 할 수 있다.

제2절 런검정

이진수열 (s_t) 에서 , 0 또는 1 이 처음부터 끝까지 반복해서 연이어 나타나는 부분을 이 이진수열의 런(run) 이라고 한다.

예를 들어, 0110001은 한 개의 0 으로 이루어진 런과 두 개의 1로 이루어진 런, 세 개의 0 으로 이루어진 런, 그리고 한 개의 1 로 이루어진 런을 포함하고 있다.

특히, 0 만으로 이루어진 런을 그 수열의 갭(gap) 이라 하고, 1 만으로 이루어진 런을 그 수열의 블록(block) 이라 한다.

다음은 주기가 N 인 이진 수열의 임의성에 관한 Golomb의 공리계이다.

R1 먼저 N이 짝수인 경우에, 길이가 N인 순환마디에는 $\frac{N}{2}$개의 0과 $\frac{N}{2}$개의 1이 들어 있다. 한편, N이 홀수인 경우에는 0 또는 1이 $[\frac{N}{2}]$개씩 들어 있다.

R2 길이가 N인 순환마디에 들어 있는 런 가운데 절반은 길이가 1 이고 $\frac{1}{4}$은 길이가 2 이며, 일반적으로 이 순환마디에 적어도 2^{i+1} 개의 런이 들어 있다면 이 중에서 $\frac{1}{2^i}$ 은 길이가 i 이다. 또한, 각 $i>1$ 에 대하여 길이가 i 인 갭 과 블록의 개수는 동일하다.

R3 이 수열의 out-of-phase 자기상관은 일정하다.

위의 R1, R2, R3 를 만족시키는 무한 이진수열을 흔히 G-random 수열 또는 PN 수열 이라고 한다.

이진수열 (s_t) 의 N 개의 항 $(s_0, s_1, s_2, , \cdots, s_{m-1})$ 에서 갭의 개수와 블록의 개수를 각각 r_0, r_1이라 하고 $n = r_0 + r_1$이라고 하자. 또, 각 $i(1 \le i \le L)$에 대하여 길이가 i 인 갭의 개수와 길이가 i 인 블록의 개수를 각각 n_{0i}, n_{1i} 라 하고 $n_i = n_{0i} + n_{1i}$ 라고 하면, n 은 런의 개수이고, n_i 는 길이가 i 인 런의 개수이다.

이때, 갭 중에서 그 길이가 i 일 확률은 $\frac{1}{2^{i+2}}$ 이고 블록 중에서 그 길이가 i 일 확률은 $\frac{1}{2^{i+2}}$ 이다. 따라서 통계량

$$T = \sum_{j=0}^{1} \sum_{i=1}^{L} \frac{\left(n_i^k - \frac{n}{2^{i+2}} \right)^2}{\frac{n}{2^{i+2}}}$$

은 근사적으로 자유도가 $2L$인 카이제곱분포를 따른다. 이와 같은 통계량을 이용하

는 검정법을 매개변수가 L인 런 검정법이라고 한다.

예를 보면 다음과 같다.

입력 e = 1100100100001111110110101010001000100001011010001 1
0000100011010011000100110001100110001010001011 1000

n = 100

t = 0.02

출력 p = 0.42

Vn(obs) = 52

p-value = 0.500798

결과 : p-value > 0.01 이므로 이 수열은 랜덤하다고 볼 수 있다.

제3절 포커검정

이진수열 (s_t)의 N 개의 항 $s_0, s_1, s_2, \cdots, s_{N-1}$ 을 처음부터 차례로 m개씩 나누면, 다음과 같이 m 차원 벡터공간 $GF(2)^n$ 에 속하는 $n = [\frac{N}{m}]$개의 벡터가 생긴다.

$$
\begin{array}{ll}
 & (s_0,\ s_1, \cdots,\ s_{t+2},\ \cdots, s_{m-1}) \\
* & (s_m,\ s_{m+1},\ s_{m+2},\ , \cdots, s_{2m-1}) \\
 & \cdots\cdots\cdots\cdots\cdots\cdots \\
 & (s_{(n-1)m},\ s_{(n-1)m+1}\ , \cdots, s_{nm})
\end{array}
$$

이들 벡터 중에서 i 개의 1과 $m-1$개의 0으로 이루어진 벡터의 개수를 $n(i)$라고 하자. 이 때, 수열 (s_t) 가 랜덤하다면, 각 $i(1 \le i \le m)$에 대하여 $n(i)$의 평균값은 $\binom{m}{i}$ 이다.

따라서 통계량

$$T = \sum_{i=0}^{m} \frac{\left(n(i) - \binom{m}{i} \frac{n}{2^m} \right)^2}{\binom{m}{i} \frac{n}{2^m}} = \frac{2^m}{n} \sum_{i=0}^{m} \frac{n(i)^2}{\binom{m}{i}} - n$$

는 근사적으로 자유도가 m 인 χ^2 분포를 따른다.

이와 같은 난수성 검정법을 포커검정(poker test) 이라고 한다.

이 검정법은 프리퀀시 엠검정법에서 사용되는 통계량보다 단순하므로 이용하기에는 쉬우나 정밀성은 떨어진다.

제4절 시리얼 검정

이 검정법은 한 항이 그 다음에 0 또는 1 로 바뀌어 나가는 과정이 랜덤 한지를 조사하는 방법이다.

이진 수열 (s_t) 의 N 개의 항 $s_0, s_1, s_2, \cdots, s_{N-1}$ 중에서 0 인 것의 개수와 1인 것의 개수를 각각 n_0, n_1 이라고 하자. 또 이들 N 개의 항을 차례를 연이어 두 항씩 묶어 놓은

$$s_0 s_1,\ s_1 s_2,\ s_2 s_3,\ \cdots,\ s_{N-2} s_{N-1}$$

중에서 00, 01, 10, 11 과 같은 것의 개수를 각각 n_{00}, n_{01}, n_{10}, n_{11} 이라고 하자.

이때, 다음 등식이 성립한다.

$$n_0 + n_1 = N$$

$$n_{00} + n_{01} = n_0 \text{ 또는 } n_0 - 1$$

$$n_{10} + n_{11} = n_1 \text{ 또는 } n_1 - 1$$

$$n_{00} + n_{01} + n_{10} + n_{11} = N-1$$

한편, 각 n_{ij} 의 기댓값은 $\frac{N-1}{4}$ 이다. 따라서 통계량

$$T = \sum_{i,j=0}^{1} \frac{\left(n_{ij} - \frac{N-1}{4}\right)^2}{\frac{N-1}{4}} - \sum_{i=0}^{1} \frac{\left(n_i - \frac{N}{2}\right)^2}{\frac{N}{2}}$$

는 근사적으로 자유도가 2 인 χ^2 분포를 따른다.

위 식은 다음과 같이 간단히 된다.

$$T = \frac{4}{n-1}(n_{00}^2 + n_{01}^2 + n_{10}^2 + n_{11}^2) - \frac{2}{N}(n_0^2 + n_1^2) + 1$$

이 통계량을 이용하여 난수성을 검정하는 방법을 시리얼 검정법(serial test)이라고 한다.

예를 들어 길이 10인 이진수열 0011011101 에 대하여 비트길이를 3까지로 제안한다면 3비트씩 나누어야 하므로 끝에 00를 첨가한 001101110100을 이용하여 빈도를 측정한다.

같은 방법으로 2비트열에 대한 빈도 측정은 00110111010을 사용하고 단일비트에 대한 빈도수 측정은 프리퀀시검정과 동일한 방법으로 시행한다.

각 비트열에 대한 빈도수 측정에 대한 확률값은 다음과 같은 식에 의해 산출된다.

n : 총도수, m : 비트열수(여기서는 3), v : 각 비트의 패턴

3비트 : $\Psi_m^2 = \frac{2^m}{n} \sum_{i_1 \cdots i_m} \left(v_{i_1 \cdots i_m} - \frac{n}{2^m} \right)^2 = \frac{2^m}{n} \sum_{i_1 \cdots i_m} v^2_{i_1 \cdots i_m} - n$

2비트 : $\Psi_{m-1}^2 = \frac{2^{m-1}}{n} \sum_{i_1 \cdots i_{m-1}} \left(v_{i_1 \cdots i_{m-1}} - \frac{n}{2^{m-1}} \right)^2 = \frac{2^{m-1}}{n} \sum_{i_1 \cdots i_{m-1}} v^2_{i_1 \cdots i_{m-1}} - n$

1비트 :

$$\Psi_{m-2}^2 = \frac{2^{m-2}}{n} \sum_{i_1 \cdots i_{m-2}} \left(v_{i_1 \cdots i_{m-2}} - \frac{n}{2^{m-2}} \right)^2 = \frac{2^{m-2}}{n} \sum_{i_1 \cdots i_{m-2}} v^2_{i_1 \cdots i_{m-2}} - n$$

를 이용하면

$$\Psi_3^2 = \frac{2^3}{10}(0+1+1+4+1+4+4+1) - 10 = 12.8 - 10 = 2.8$$

$$\Psi_2^2 = \frac{2^2}{10}(1+9+9+9) - 10 = 1.2$$

$$\Psi_1^2 = \frac{2}{10}(16+36) - 10 = 10.4 = 0.4$$

여기서, χ^2 분포에서 유의수준 5%, 자유도 2의 한계값은 5.99 이다.

따라서 T의 통계량의 값이 5.99 보다 큰 경우에, 시리얼검정에 대하여 유의수준 5% 로 이 이진수열은 난수성이 없다고 판단되어 기각한다.

제5장 제안하는 난수성 검정 방법

제안하는 의사난수 생성기에 의한 출력물에 대한 난수성 테스트는 먼저 난수의 기본 성질인 예측불가능성에 대한 안전도를 확보하는데 주안점을 두었다. 난수성을 만족하는 요소로써 구별불능성이 있고 선행비트들에 대한 후위 비트의 예측을 할 수 없어야 하며 기존에 나와 있는 트래픽으로부터 이후 출력값들의 분석을 할 수 없어야 한다. 또한 주어진 난수열의 주기가 판별할 수 없어야 한다.

이를 실험적으로 검증하기에는 무리가 따르기에 확률적 방법을 이용하여 난수성에 근접하는 데이터에 대한 신뢰도를 인증한다.

기존에 나와 있는 의사난수 테스트들 중에서 프리퀀시검정, 런검정, 시리얼검정들은 출력 비트들에 빈도수를 측정하여 이를 통계적 분포도로 산출한다는 비슷한 특징을 지니고 있다. 이에 본고에서 제안하는 의사난수 생성기에 의한 출력 데이터의 통계적 테스트는 프리퀀시, 런, 시리얼 검정의 압축된 형태의 구현방식으로 난수성에 대한 검증을 진행하였다.

제안하는 통계적 난수테스트는 선행 비트들의 출력에 대한 후위비트의 형태양상을 빈도수를 통한 측정으로 구한다음 통계적 분포도로써 난수성 검증을 완성한다.

사용 예는 다음과 같다.

이진 비트열 122비트 A에서 각각의 비트열의 형태를 파악하여 이후 나오는 비트의 상태를 평가한다.

A=1101010100000001010011100100101101100000111011110010101110101000101101100010000100011011010000000110001010111010001100100 0

에 대하여 선행 비트 1이 나온 후 이후 다시 1이 나올 비트들의 수를 a1이라하고 선행비트 0이 나온 후 다시 1이 나올 경우를 a0 이라 하면

$$a1 = 20$$

$$a0 = 32$$

이는 시리얼 테스트의 m-2에 해당하는 2비트 데이터의 빈도수에 대한 검정에서

11, 10 의 빈도수만을 측정하는 것과 마찬가지라 볼 수 있다. 또한 선행비트 10, 11, 01, 00 이 나온 후 다시 이들 후에 1이 나올 경우의 수를 계산해 보면 다음과 같다.

a11 = 6

a10 = 18

a01 = 13

a00 = 14

이는 시리얼테스트의 m-3에 해당하는 테스트들 중에서 111, 101, 011, 001 만의 빈도수를 측정하는 것과 마찬가지이다. 또한 여기서 111에 대한 빈도수의 측정은 런(run)테스트의 3단 진행을 의미하기도 하므로 본 테스트는 런 테스트와 시리얼 테스트와 밀접한 관계가 있음을 알 수 있다.

이를 시리얼 테스트의 Ψ 함수에 적용하면

$$\Psi_m^2 = \frac{\frac{2^m}{2}}{n} \sum_{i_1 \cdots i_m} \left(v_{i_1 \cdots i_m} - \frac{n}{\frac{2^m}{2}} \right)^2$$

3비트 : $$= \frac{2^{m-1}}{n} \sum_{i_1 \cdots i_m} \left(v^2_{i_1 \cdots i_m} - \frac{n}{2^m} \right)^2$$

$$= \frac{2^{m-1}}{n} \sum_{i_1 \cdots i_m} v^2_{i_1 \cdots i_m} - n$$

$$\Psi_{m-1}^2 = \frac{\frac{2^{m-1}}{2}}{n} \sum_{i_1 \cdots i_{m-1}} \left(v_{i_1 \cdots i_{m-1}} - \frac{n}{\frac{2^{m-1}}{2}} \right)^2$$

2비트 : $$= \frac{2^{m-2}}{n} \sum_{i_1 \cdots i_{m-1}} \left(v_{i_1 \cdots i_{m-1}} - \frac{n}{2^{m-2}} \right)^2$$

$$= \frac{2^{m-2}}{n} \sum_{i_1 \cdots i_{m-1}} v^2_{i_1 \cdots i_{m-1}} - n$$

$$\Psi^2_{m-2} = \frac{\frac{2^{m-2}}{2}}{n} \sum_{i_1 \cdots i_{m-2}} \left(v_{i_1 \cdots i_{m-2}} - \frac{n}{\frac{2^{m-2}}{2}} \right)^2$$

1비트 : $$= \frac{2^{m-3}}{n} \sum_{i_1 \cdots m-2} \left(v_{i_1 \cdots i_{m-2}} - \frac{n}{2^{m-3}} \right)^2$$

$$= \frac{2^{m-2}}{n} \sum_{i_1 \cdots i_{m-2}} v^2_{i_1 \cdots i_{m-2}} - n$$

과 같이 나타낼 수 있다.

이는 시리얼 테스트에서 한단계 아래값들의 연산 즉 Ψ^2_{m+1}, Ψ^2_m, Ψ^2_{m-1}의 값과 같음을 알 수 있다. 다시 말하여 앞비트에 의한 후위 비트의 빈도수를 측정하는 것은 전위 데이터의 영향을 받으므로 통계적 결과치는 전위 데이터에 종속되며 검정량은 $m+1$ 비트의 검정을 시행한 것과 동일한 효과를 볼 수 있다.

이를 적용 시에는 3비트 연산을 시행하면 이전 2비트까지의 시리얼검정을 정확히 통과했다는 가정아래 3비트형의 데이터 분포에서 $\frac{1}{2}$가지의 시행만 하면 된다.

탐색알고리즘에 있어서 n 비트의 데이터에 대하여 작은 블럭인 m 비트열로 나눈다면 이에 대한 검색 타임은 $\frac{m}{3} \times \frac{8}{2} \times \frac{n}{m} = \frac{4}{3} \times n$ 회의 타임 절약이 발생한다. 보편 적으로 $n = 100000$ 비트 이상의 데이터에 대한 난수성을 테스트 하므로 절약 되는 시간은 약 1천3백만 타임의 탐색을 절약할 수 있다.

이를 일반화 시켜서 최종 8비트까지의 절약되는 연산수를 계산하면 다음과 같다.

t : 최종 비트수

n : 랜덤 스트링

m : 블록길이

$$\frac{m}{t} \times \frac{2^t}{2} \times \frac{n}{m} = \frac{1}{t} \times 2^{t-1} \times n \text{ 이므로}$$

n = 100000

t=3	$\frac{4}{3} \times n$	=133333	t=4	$\frac{8}{4} \times n$	=200000
t=5	$\frac{16}{5} \times n$	=320000	t=6	$\frac{32}{6} \times n$	=533333
t=7	$\frac{64}{7} \times n$	=914285	t=8	$\frac{128}{8} \times n$	=1600000

참고문헌

[1] M. Akkar and C. Giraud, "An Implementation of DES and AES, Secure against Some Attacks", CHES 2001, LNCS 2162, pp. 309-318, 2001.

[2] ChaeHoon. Lim "A Revised Version of CRYPTON" Information and Communications Research Center.

[3] J.Daemen, L.Knudsen and V. Rijmen, "The block cipher Square", In Fast Software Encryption, Lecture Notes in Computer Science(LNCS) 1267, Springer-Verlag, 1997.

[4] P. Dusart, G. Letourneux, and O. Vivolo, "Differential Fault Analysis on A.E.S", http://eprint.iacr.org/2003/010.pdf

[5] H. Feistel, "Block Cipher Cryptographic System," U.S. Patent #3,798,359,19 Mar 1974.

[6] N.Goots, B. Izotov, A. Moldovyan, and N. Moldovyan, "Modern Cryptography: Protect Your Data with Fast Block Ciphers", A-LIST Publishing, 2003.

[7] L. R. Knudsen, "Block Ciphers-Analysis, Design and Applications," Ph.D Thesis, Computer Science department, Aarhus University, 1994.

[8] A. Menezes, P. van Oorschot, S. Vanstone, "Handbook of Applied Cryptography", CRC Press, 1996.

[9] B. Schneier, "Applied Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C (cloth)" 1996.

[10] B. Schneier, "The uses and Abuses of Biometrics,"Communications of the ACM, Vol. 42, No, 8, p136, 1999.

[11] D. Stinson, "Cryptography: Theory and Practice" CRC Press, 1995

[12] NIST, "Data Encryption Standard(DES)", http://www.itl.nist.gov/fipspubs/fip46-2.htm

[13] NIST, "Instruction-level Parallelism in AES Candidates", http://cs-www.ncsl.nist.gov/CryptoToolkit/aes/round1/conf2/papers/clapp.pdf.

[14] NIST, "Federal Information Processing Standards Publication 197-Specification for the Advanced Encryption Standard (AES)", http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf , 2001.