비밀키 암호 시스템의 통계적 특성 분석을 위한 **Randomness test** 방법의 비교 고찰

김 종희, 염 대현, 이 필중
포항공과대학교 전자전기공학과

On Randomness tests for the Statistical Analysis of Symmetric Ciphers

Chong Hee Kim, Dae Hyun Youm, Pil Joong Lee
Dept. of Electronics and Electrical Engineering, POSTECH
chhkim@oberon.postech.ac.kr, daehyun@oberon.postech.ac.kr, pjl@postech.ac.kr

요약

본 논문에서는 스트림 암호 알고리즘과 블록 암호 알고리즘과 같은 비밀키 암호 시스템의 통계적 특성을 측정하기 위하여 사용된 여러 가지 randomness 테스트 방법들을 구현하여 그 성능을 randomness test의 power 측면에서 서로 비교하였다. 여기서, power란 randomness test가 nonrandom한 비트 스트림을 얼마나 정확하게 검정할 수 있는 가를 나타내는 척도이다. 그리고, 스트림 암호 알고리즘과 블록 암호 알고리즘의 통계적 특성을 측정하기 위해 가장 효율적이라고 생각되는 테스트 방법들을 찾아 이 방법들을 사용할 것을 제안한다. 끝으로 제안된 방법들을 이용하여, DES, AES 후보 알고리즘, SEED의 통계적 특성을 분석하였다.

1. 서론

비밀키 암호 시스템은 블록 암호 알고리즘(block cipher)과 스트림 암호 알고리즘(stream cipher)으로 나눌 수가 있다. 블록 암호 알고리즘은 암호화하고자 하는 메시지를 같은 크기의 블록으로 나누어서 각각 암호화를 시키는데 반해, 스트림 암호 알고리즘은 메시지를 비트 단위로 키 스트림과 modulo-two addition을 하여 암호화시킨다.

스트림 암호 알고리즘의 안전도는 암호화를 위해 사용하는 키 스트림이 얼마나 random한 가에 달려있다. 만일 키 스트림이 random하지 않다면 공격자가 키 스트림을 추측할 수 있게 된다. 따라서, 키 스트림의 randomness를 측정하는 것이 스트림 암호 알고리즘의 안전도를 측정하는 방법이 된다.

블록 암호 알고리즘의 경우에는 스트림 암호 알고리즘의 경우와 같이 비트 스트림의 randomness를 측정하는 방법 자체만으로는 안전도를 나타낼 수 없다. 따라서, 블록 암호 알고리즘의 안전도를 측정하기 위한 한 방법으로 평문과 암호문 사이의 독립성을 생각할

수 있다[13]. 블록 암호 알고리즘이 갖추어야 할 또 다른 통계적 특성으로 avalanche 효과를 생각할 수 있는데, 블록 암호 알고리즘의 암호문의 각 비트들이 평문의 각 비트들을 입력으로 하는 boolean function 이라고 생각하였을 때, 평문의 각 비트들이 암호문의 각 비트들에 골고루 영향을 주는 가를 측정하는 것이다.

본 논문에서는 지금까지 널리 알려진 randomness test 방법들을 비교 분석하여 스트림 암호 알고리즘과 블록 암호 알고리즘의 통계적 특성 분석을 위해 효율적인 방법을 찾아 이 방법들을 사용할 것을 제안한다. 그리고, DES[21]와 AES 후보 알고리즘[22][23]과 국내 표준 암호 알고리즘으로 제안하기 위해 개발된 SEED[19]에 대해 통계적 특성을 분석한다.

이에 2 장에서는 randomness 테스트의 성능을 비교하기 위하여 power function 의 개념을 소개하고, 3 장에서는 지금까지 널리 알려진 randomness 테스트 방법들에 대해 소개한다. 그리고, 4 장에서는 randomness test 방법들의 power function 을 비교하여 스트림 암호 알고리즘과 블록 암호 알고리즘의 통계적 특성을 측정하기 위해 가장 효율적이라고 생각되는 테스트 방법들을 찾아 이 방법들을 사용할 것을 제안한다. 5 장에서는 DES, AES 후보 알고리즘, SEED 의 통계적 특성을 분석하고, 마지막으로 6 장에서는 결론을 맺는다.

2. Power function

만일 randomness test 에서 통계량이 기각역(critical region)에 떨어지게 되면 주어진 영 가정(null hypothesis)은 기각이 되게 되고, alternative hypothesis 가 받아들여지게 된다. 그러나, 영 가정이 사실이라고 할지라도 기각역에 떨어질 수가 있고, 이러한 경우를 Type I error 라 한다. 이 때의 확률을 α라고 한다. 이와는 반대로 통계량이 기각역이 아닌 곳에 떨어져서 영 가정이 받아들여지더라도 실제로는 영 가정이 사실이 아닐 수가 있다. 이러한 경우를 Type II error 라고 하며 이 때의 확률을 β라고 한다.

Randomness test 의 power 란 영 가정 H_0가 사실이 아닐 때 즉, alternative hypothesis H_A가 사실일 때 영 가정을 기각할 수 있는 확률이다[7]. 즉, Power 는 $1-\beta$가 되며, [0,1]의 값을 가진다. Randomness test 의 power 를 이용하여 주어진 randomness test 가 alternative hypothesis 를 따르는 비트 스트림을 얼마나 정확하게 찾아낼 수 있는 가를 알 수 있다. 비트 스트림의 크기 n 이 일정하고 유의 수준 α가 주어졌을 때, randomness test 의 power 는 통계량이 영 가정으로부터 벗어날 수록 커지게 된다. 또, 비트 스트림의 크기 n 이 증가할 수록 power 는 증가하게 된다. 따라서, power function 을 이용하여 여러 가지 randomness test 들을 비교할 수 있다.

본 논문의 대부분의 randomness test 에서 사용한 영 가정은 다음과 같다.

- H_0 : 비트 스트림 S 는 각 비트들이 $Pr(s_i = 1) = p = 1/2$ 인 동일한 분포를 가지고, 서로 독립적인 n 개의 이진 random variable s_i 들로 구성되어 있다.

Non-randomness 를 측정하기 위하여 다음과 같은 두 가지의 alternative hypothesis 를 본 논문에서는 사용하였다.

- H_1 : 비트 스트림 S 는 각 비트들이 $\mathbf{Pr(s_i = 1) = p \neq 1/2}$ 인 동일한 분포를 가지고, 서로 독립적인 n 개의 이진 random variable s_i 들로 구성되어 있다.
- H_2 : 비트 스트림 S 는 각 비트들이 $Pr(s_i = 1) = p = 1/2$ 인 동일한 분포를 가지고, 서로 **독립적이지 않은**$\mathbf{(Pr(s_i = 1 | s_{i-1} = 0) = Pr(s_i = 0 | s_{i-1} = 1) = q \neq 1/2)}$ n 개의 이진 random variable s_i 들로 구성되어 있다.

| p - 1/2 |과 | q - 1/2 |의 값이 증가할수록 randomness test 의 power 는 증가하게 되고, 이렇게 증가하는 정도가 클수록 nonrandom 비트 스트림을 잘 검정하는 테스트라고 할 수 있다.

Randomness test 의 power function 을 실험적으로 구하기 위해서 본 논문에서는 위에서 설명한 두 가지 alternative hypothesis H_1, H_2를 따르는 비트 스트림을 사용하였다. 각각의 p 의 값과 q 의 값에 대해 각각 10,000 개의 비트 스트림을 생성하였다. 이러한 비트 스트림을 생성하기 위하여 combined linear congruential generator 을 사용하였다[3]. 그리고, 기각역을 설정하기 위해 유의 수준 5%, 1%를 사용하였다.

주어진 p (또는 q)에 해당하는 길이 n 인 비트 스트림 10,000 개를 독립적이라 가정하고, 각각에 대해 통계량을 계산한다. 그리고, 이 통계량들 가운데 기각역에 떨어지는 것들의 개수를 전체 샘플의 개수 10,000 으로 나눈 값을 구하면 이 값이 주어진 randomness test 의 확률 p (또는 q)에 해당하는 power function 의 값이 된다. Power function 은 p = 1/2, q = 1/2 에 대해 대칭적이므로 p, q > 1/2 일 때의 power function 의 값은 p, q < 1/2 일 때의 값과 같게 된다.

3. Randomness tests 방법

본 장에서는 지금까지 알려진 여러 가지 randomness test 방법들에 대해 간단히 소개한다.

3.1 Frequency test

길이가 n 인 비트 스트림에서 0 과 1 의 개수가 균일하게 분포되어 있는 가를 검정하는 방법으로 uniformity test, equidistribution test 라고도 한다. n_1 을 비트 스트림의 1 의 개수라고 하면 $z = 2\sqrt{n}\left(\frac{n_1}{n} - \frac{1}{2}\right)$는 표준정규분포 N(0,1)을 따른다[5][7]. 비트 스트림을 길이가 b(b | n, 1 < b < n)인 subblock 들로 나눌 경우, 각각의 subblock 의 발생빈도가 동일한 가를 검정하는 방법도 있다[7].

3.2 Runs test

비트 스트림에서 run 의 개수나 분포가 random 한가를 검정하는 방법이다. 여기서, run 이란 비트 스트림에서 동일한 값이 연속해서 있는 경우를 말한다. Run 에는 1 로 이루어진 경우(block)와 0 으로 이루어진 경우(gap)가 있으며, 비트 스트림에서 전체 run 의 개수가 random 한 가를 테스트하는 runs test 와 비트 스트림에서 각각의 길이에 해당하는 run 의 분포가 random 한가를 테스트하는 runs distribution test 로 나눌 수 있다.

3.2.1 Runs test

길이 n 인 비트 스트림에 대해 R 을 전체 run 의 개수라고 하면

$$z = \frac{R - \frac{n+1}{2}}{\sqrt{\frac{n-1}{4}}} = \frac{2R - n - 1}{\sqrt{n-1}}$$

는 표준정규분포 N(0,1)을 따른다[4][7].

3.2.2 Runs distribution test

방법 1

길이가 i 인 run 에 대해 1 의 run(block)의 개수와 0 의 run(gap)의 개수를 각각 B_i, G_i라고 하면, 길이 i 인 block(gap)의 개수의 기대값 e_i는

$$e_i = \frac{\frac{n-i-1}{2}+2}{2^{i+1}}$$

이고[12],

$$\chi^2 = \sum_{i=1}^{k}\frac{(B_i - e_i)^2}{e_i} + \sum_{i=1}^{k}\frac{(G_i - e_i)^2}{e_i}$$

은 자유도 2k-2 인 $\chi 2$ 분포를 따른다[5].

방법 2

방법 1 과 같으나 기대값으로 $e_i = \frac{n}{2^{i+2}}$를 사용한다[1].

방법 3

방법 2 와 같으나, Block 과 Gap 중 하나만을 선택하여 다음과 같은 통계량에 대해 자유도 16 인 χ^2 test 를 한다[24].

$$\chi^2 = \sum_{i=1}^{17}\frac{(B_i - e_i)^2}{e_i}$$

여기서는 runs 의 길이가 17 이상이 나올 확률은 거의 없다고 간주하여 17 이상의 길이를 가진 runs 을 하나의 그룹으로 설정하였다[24].

3.3 Change point test

비트 스트림에서 t 번째 비트에 대해 그 이전의 1 의 비율과 그 이후의 1 의 비율을 비교하여 그 차이가 최대인 점을 찾는 방법이다[5][7]. 이 테스트의 가정은 전체 스트림에서 1 의 비율에 변화가 없다는 것이다. 이 가설을 검정하기 위한 통계량은 $U[t] = nS[t] - tS[n]$ 이다[14]. 여기서, t = 1, ..., n-1 이고, n 은 전체 스트림의 비트 수이고, S[n]은 스트림의 전체 1 의 개수이고, S[t]는 t 비트까지의 1 의 개수이다.

U[t]의 절대값 중 최대인 것을 M 이라고 하면 즉, $M = Max|U[t]|$이면 tail area probability α는 $\alpha = e^{-\frac{2M^2}{nS[n](n-S[n])}}$ 가 된다[14].

이 α의 값은 단측 검정에 적용되는 값이므로, 양측 검정에 적용할 때는 2α를 사용한다.

3.4 Autocorrelation test

비트 스트림에서 일정 거리만큼 떨어진 비트들 간의 상관성(correlation or dependence)을 검정하는 방법이다.

방법 1

길이 n인 비트 스트림 S의 각 비트들을 s_i라고 할 때, 새로운 스트림 X를 다음과 같이 정의한다.

$X=\{x_i \mid x_i = s_i \oplus s_{i+d}\}$

X가 i.i.d.(independently and identically distributed)하다고 가정하면 §3.1에서 설명한 frequency test를 적용할 수 있다. 따라서, 통계량 $z_d = \frac{2}{\sqrt{M}}\left(A_d - \frac{M}{2}\right)$, $d = \left\lfloor \frac{1}{4}n \right\rfloor, \ldots, \left\lfloor \frac{3}{4}n \right\rfloor$ 는 표준정규분포 N(0,1)을 따르게 된다[7][1]. 여기서, $A_d = \sum_{i=1}^{\min(d,n-d)} s_i \oplus s_{i+d}$ 이다. X의 각 비트들간의 독립성(Independence)을 가정하기 위해 S의 각 비트들은 한 번씩만 사용한다. 따라서, 거리가 d일 때, X의 비트 수는 min(d, n-d)가 된다. d = n/2 일 때, X의 크기는 최대값 n/2 이 된다.

방법 2

$(s_0, s_d), (s_1, s_{d+1}), \ldots, (s_{n-1-d}, s_{n-1})$에 대해 §3.7의 2-serial test를 한다[1].

방법 3

길이 n인 비트 스트림 S의 각 비트들을 s_i라고 할 때, $A_d = \sum_{i=0}^{n-d-1}(1-2s_i)(1-2s_{d+i})$ 는 평균이 0이 되고 분산이 n-d가 된다[1], n-d가 충분히 클 경우(≥ 30)에는 통계량 $z = \frac{A_d}{\sqrt{n-d}}$ 는 표준 정규분포 N(0,1)을 따른다.

방법 4

다음과 같은 통계량 A_d에 대해

$$A_d = \sum_{i=0}^{n-1-d} s_i s_{i+d}, \quad 0 \leq d \leq n-1$$

0의 개수를 n_0, 1의 개수를 n_1이라고 두었을 때, $s_i s_{i+d}$의 기대값은 $\left(\frac{n_1}{n}\right)^2$가 되고, A_d의 기대값은 $e_d = \left(\frac{n_1}{n}\right)^2 (n-d)$가 된다[1][24]. 이 기대값을 이용해 chi-square test를 한다.

3.5 Poker test

Hamming weight test라고도 하며, 길이가 n인 전체 비트 스트림을 길이가 b인 subblock

들로 나누었을 때, 각 subblock에 포함된 1의 개수가 균일하게 분포되어 있는 가를 테스트 하는 것이다[1][7]. 다음과 같은 기대값을 갖는 자유도 b인 chi-square 테스트를 한다.

$$e_i = \binom{b}{i} 2^{-b} \times S, i = 0,1,...,b$$

여기서, S는 subblock들의 개수를 나타낸다. 즉, $\frac{n}{b} = S$이다.

3.6 Universal test

Universal source coding algorithm[17][18]에서 응용한 것으로 비트 스트림의 압축율를 이용하여 테스트하는 방법이다[7][5]. 실제로는 비트 스트림을 압축하는 대신 압축된 비트 스트림의 길이를 나타낼 수 있는 값(a measure of entropy)을 사용하여 측정한다[11].

이 값을 측정하기 위해 비트 스트림을 길이가 b인 subblock들로 나누어야 하며 따라서, 전체 비트 스트림의 길이 n = (Q+K)b 비트가 된다. Q는 initialization block들의 개수이고, K는 테스트할 block의 개수이다. Q block들에서 Tab 위치를 정하게 되는데, Tab 위치는 각각의 b비트 패턴이 마지막으로 발생한 위치가 된다. 테스트를 위한 통계량은

$$f_b = \frac{1}{K}\left[\sum_{i=Q+1}^{Q+K} \log_2 (i - Tab[s(i)])\right]$$

이다. 여기서, Tab[s(i)]는 b비트 패턴 s(i)가 마지막으로 발생했던 위치이다. 이 테스트를 하기 위한 최소 길이는 $Q = 10 \times 2^b$, $K = 1000 \times 2^b$ 이다[11]

Q와 K가 위의 조건을 만족할 경우, 표준정규분포의 통계량 z는

$$z = \frac{f_b - E_b}{c(b,K)\sqrt{Var(f_b)}}$$ 가 된다.

E_b와 $Var(f_b)$는 f_b의 평균과 분산이고, $c(b,K) = 0.7 - \frac{0.8}{b} + (1.6 + \frac{12.8}{b}) K^{-\frac{4}{b}}$ 이다[11].

3.7 Serial test

Serial test는 길이가 b인 overlapping subblock들에 대해 2^b의 패턴들이 균일하게 분포되어 있는 가를 검정하는 것이다[4]. Good은 근사적으로 chi-square 분포를 따르는 표 1과 같은 통계량을 정의하였다[6].

통계량	자유도	조건
1. $\varphi_b^2 - \varphi_{b-1}^2$	2^{b-1}	$b \geq 1$
2. $\varphi_b^2 - 2\varphi_{b-1}^2 + \varphi_{b-2}^2$	2^{b-2}	$b \geq 2$

표 1 : Serial test를 위한 Good의 통계량

표 1에서 $\varphi_b^2 = \frac{2^b}{n-b+1} \sum_{t=0}^{2^b-1} \left(n_t - \frac{n-b+1}{2^b} \right)^2$ 이고, $\varphi_0^2 = 0$이다. Subblock으로 나누는 길이 b

에 따라 b-serial test 라고 한다.

3.8 Binary derivative test

길이 n 인 비트 스트림의 binary derivative 란 연속하는 두 비트를 modulo-two addition 하여 생성되는 새로운 길이 n-1 의 비트 스트림을 말한다[20].

예를 들면 비트 스트림 S 에서, 첫번째 binary derivative $d_1(S)$는 다음과 같다.

S = 0 1 1 0 1 0 0 0 1 0 0 1 1 1

$d_1(S)$ = 1 0 1 1 1 0 0 1 1 0 1 0 0

여기서, $d_k(S)$를 S 의 k 번째 binary derivative 라고 한다.

각각의 Binary derivative 와 원래의 비트 스트림에 대해 frequency test 를 적용한다.

3.9 Linear complexity test

비트 스트림 S 를 만들어 내기 위해 필요한 LFSR(linear feedback shift register)의 최소 길이를 Linear complexity L(S)라고 한다[7]. 만일 L(s)의 값이 L 이라면, 2L 의 연속적인 비트들로 Berlekamp-Massey 알고리즘을 이용해 모든 비트 스트림을 재구성할 수 있다[10]. 따라서, 비트 스트림이 재구성되지 않게 하기 위해서는 L 의 값이 커야만 한다. 스트림 암호 알고리즘에서는 키 스트림의 linear complexity 가 낮은 경우 안전도에 중요한 영향을 주게 되므로 linear complexity 를 검사하는 것은 중요하다고 할 수 있다.

예를 들어 다음과 같은 비트 스트림은

0101100101010010011110000011011100110001110101111110 1101

다음과 같은 관계식으로 표현될 수 있다.

$$u(t+6) = u(t+5) \oplus u(t+4) \oplus u(t+1) \oplus u(t)$$

따라서, 이 비트 스트림의 Linear complexity 는 6 이 된다. Random 한 비트 스트림의 경우, 길이 n 이 클 경우 L(s) 는 평균이 $\frac{n}{2}$ 이고, 분산이 $\frac{86}{81}$ 인 정규분포를 따른다[15]. 따라서, 통계량 $z = \sqrt{\frac{81}{86}}\left(L(s) - \frac{n}{2}\right)$ 는 표준정규분포 N(0,1)을 따르게 된다[15].

위에서 설명한 Linear complexity test 에서 random 하다고 평가가 된 비트 스트림이 highly patterned 이거나 또는 highly patterned 된 큰 substring 을 포함하고 있을 수가 있다. 따라서, linear complexity profile 의 변화를 측정할 필요가 있다. S(i)를 비트 스트림 S 의 처음부터 i 번째 비트까지로 구성된 substring 이라고 하자. 그리고, L(S(i))를 S(i)의 linear complexity 라고 하면, L(S(i))의 값들을 S 의 linear complexity profile 이라고 정의하고[10], 이 linear complexity profile 은 $\frac{i}{2}$ line 을 따르게 된다[10].

만일, L(S(i)) > L(S(i-1))이면 jump 가 발생했다고 한다. 길이 n 인 비트 스트림에서 jump 의 개수를 F 라고 하면, n 이 클 때 F 는 근사적으로 평균이 $\frac{n}{4}$ 이고, 분산이 $\frac{n}{8}$ 이 된다. 따라서, 통계량 $z = \sqrt{\frac{8}{n}}\left(F - \frac{n}{4}\right)$ 는 표준정규분포 N(0,1)을 따르게 된다[2]. Jump 의 수가 적을 경우에는 스트림 내에 특정 패턴이 존재함을 의미하게 되며, 단측 검정을 한다.

또한 jump 의 크기를 생각할 수 있는데, jump 의 크기란 change 가 발생했을 때의 linear complexity 의 차이를 말한다. 즉, jump 가 발생했을 때의 L(S(i)) - L(S(i-1))이다[2].

Jump의 전체 개수를 F라고 하고, o_i를 크기가 i인 jump의 관측된 개수라고 하면, 크기가 i인 jump의 기대값 $e_i = 2^{-i} \times F$가 된다[2]. 따라서, 통계량

$$\chi^2 = \sum_{i=1}^{k} \frac{(o_i - e_i)^2}{e_i}$$

자유도 k-1인 chi-square 분포를 따른다.

블록 암호 알고리즘에서의 linear complexity test는 다음과 같이 구할 수 있다. 길이가 n인 블록 암호 알고리즘에서 Linear complexity가 L인 블록의 개수를 $N_n(L)$이라고 하면,

$$N_n(L) = \begin{cases} 2^{\min\{2n-2L,\,2L-1\}} & for\ n \geq L > 0 \\ 1 & for\ n > L = 0 \end{cases}$$

가 된다[15]

만일 모든 비트들이 0인 스트림의 linear complextiy를 0이라고 한다면, 길이가 n인 블록의 linear complexity는 0에서 n사이의 값을 가진다.

블록의 개수가 S일 때, Linear complexity가 L인 블록의 기대치는

$$e_L = N_n(L)2^{-n} \times S, \qquad \text{for } 0 \leq L \leq n$$

가 되고, 이를 이용해 Chi-square 테스트를 한다.

3.10 Sequence complexity test

비트 스트림의 처음 비트부터 순차적으로 검사할 때 새로운 형태(pattern)가 나오는 개수를 그 비트 스트림의 sequence complexity라고 한다[8][9][4]. 예를 들어 다음과 같은 2진 수열에서

1000111101100001110 = 1/0/01/1110/1100/001110

수열의 complexity는 6이 된다.

[8]에 의하면 길이가 n인 비트 스트림의 최소 complexity는 $\frac{n}{\log_2 n}$가 된다. 따라서, 이 값을 비트 스트림이 randomness을 가지고 있는 가에 대한 경계값(threshold value)으로 사용한다[4][9].

3.11 Avalanche test

블록 암호 알고리즘에서 평문의 한 비트가 바뀌었을 때 평균적으로 암호문의 비트들의 반이 바뀌었을 때 블록 암호 알고리즘은 avalanche 특성을 만족한다고 한다[16][7][5]. 또, 평문의 한 비트가 바뀌었을 때 암호문의 각각의 비트가 바뀔 확률이 1/2이 된다면 블록 암호 알고리즘은 strict avalanche 특성을 만족한다고 하며[16][7][5], 이러한 성질을 블록 암호 알고리즘이 만족하는 가를 테스트 하는 것은 중요하다고 할 수 있다.

예를 들어, 블록 암호 알고리즘의 블록 크기가 128비트이고 평문의 한 비트가 바뀌었을 때 암호문에서 바뀌어진 비트들의 개수를 random variable X라고 하자. 만일 좋은 블록 암호 알고리즘이라면 X는 Bernoulli random variable을 따르게 된다. 따라서, X의 평균은 64가 되며 표준편차는 $\frac{4\sqrt{2}}{\sqrt{N}}$가 된다[16]. 여기서 N은 X의 표본의 수가 된다. 예를 들어, a개의 임의의 키와 b개의 임의의 블록을 선택하여 블록 크기 128비트인 알

고리즘에 대해 avalanche test를 한다면 $N = a \times b \times 128$ 이 된다.

만일, 유의수준 0.3%로 검정을 하면 $|X - 64| \leq \frac{4\sqrt{2}}{\sqrt{N}} \times 3$ 일 때 avalanche 성질을 만족한다고 할 수 있다[16].

Strict avalanche 특성을 측정하기 위해 v_{ij}를 평문의 i번째 비트가 바뀌었을 때, 암호문의 j번째 비트가 바뀌면 1의 값을 가지고 그렇지 않을 경우 0의 값을 가지는 random variable이라고 하자. 전체 샘플의 개수가 S일 경우, 통계량 $z_{ij} = 2\sqrt{S}\left(a_{ij} - \frac{1}{2}\right)$는 표준정규분포 N(0,1)을 따르게 된다[7]. 여기서, $a_{ij} = \frac{1}{S}\sum_{p=1}^{S} v_{ij,p}$ 이다.

4. 실험결과의 비교 분석 및 효율적인 randomness tests 방법의 제시

본 장에서는 §3의 randomness test 방법들에 대해 power function을 구한 후 이들을 비교 분석한다. 그리고, 실험 결과를 바탕으로 여러 가지 randomness test 방법 중 비밀키 암호 시스템의 통계적 특성 분석을 위하여 효율적인 randomness test 방법들을 스트림 암호 알고리즘과 블록 암호 알고리즘에 대해서 각각 제시한다.

4.1 실험결과의 비교 분석

길이가 n인 비트 스트림이 각각의 비트들이 0과 1이 나올 확률이 같은 n번의 독립적인 Bernoulli trial로 구성되어 있는 가를 검정하는 방법인 frequency, runs, runs distribution, autocorrelation, poker, universal, serial, binary derivative test에 대해 alternative hypothesis H_1과 H_2를 따르는 길이 n = 10,240인 비트 스트림 10,000개를 사용하여 유의 수준 5%와 1%에 대해 power function을 실험적으로 구하였다.

비트 스트림을 일정한 길이의 subblock 단위로 나누어서 테스트하는 경우인 §3.1의 frequency test를 subblock에 적용하는 방법, §3.5의 poker test, §3.6의 universal test의 경우에는 subblock의 길이 b = 4, 6에 대해 각각 universal test를 적용하기 위한 최소 비트 스트림의 길이인 n = 102,400, 387,840 인 비트 스트림 1,000개를 사용하였다. §3.4의 Autocorrelation test의 경우에는 frequency test와의 비교를 위해 n = 20,480이고, 거리 d = 10,240인 비트 스트림을 사용하였다.

실험의 결과를 표 2~14에 제시하였다. 표 2와 표 3의 결과를 살펴보면 §3.1의 frequency test는 H_1을 따르는 비트 스트림은 잘 검정을 하지만, H_2를 따르는 비트 스트림을 잘 검정할 수 없음을 알 수 있다. 즉, frequency test는 비트 스트림의 균일성을 검정하기 위해서는 필요한 테스트이지만 비트들간의 독립성을 검정하기 위해서는 사용할 수 없음을 알 수 있다. 그러나, §3.2.1의 runs test의 경우에는 이와는 반대로 비트 스트림의 독립성을 검정하기 위하여 사용할 수 있음을 알 수 있다. §3.2.2의 Runs distribution test의 경우에는 방법 1과 2가 방법 3에 비해 효율적이라고 할 수 있다.

표 5의 결과를 보면 §3.4의 autocorrelation test는 H_2를 따르는 비트 스트림을 잘 검정할 수 없다. H_1을 따르는 비트 스트림을 잘 검정할 수 있는 가를 알아보기 위해 표 4의 결과를 Frequency test의 power function인 표 2와 비교하였을 때, autocorrealtion test의 방법 1과 방법 3은 frequency test보다 낮은 power를 가졌다고 할 수 있으며, 방법 2의 경

우는 frequency test 와 비슷한 power 를 가졌다고 할 수 있다. 그러나, 방법 2 의 경우는 p = 1/2 (q = 1/2)일 때 power function 의 값이 유의 수준 α보다 매우 큰 값을 나타내고 있다. 이는 통계량으로 매우 엄격한 값을 사용하고 있다는 것을 의미하기때문에 별로 좋은 테스트라고 할 수 없다. 따라서, H_1 을 따르는 비트 스트림을 잘 검정할 수 있는 가를 알아보기 위해서는 Autocorrelation test 를 사용하는 것보다 Frequency test 를 사용하는 것이 좋다고 할 수 있다.

표 4 와 표 5 에서 방법 1 과 방법 3 의 power function 을 보면 그 값이 비슷함을 알 수 있는데, 이는 방법 1 과 방법 3 의 차이점이 방법 1 의 경우에는 거리 d 인 비트 스트림간의 상관성을 찾기 위해 원래의 비트 스트림 S 의 각 비트들의 중복을 허용하지 않는 반면 방법 3 은 중복을 허용한다는 것인데, 본 논문의 실험은 frequency test 와의 비교를 위해 d = n/2 이므로 이들 값의 차이가 없기 때문이다. 따라서, 방법 1 과 방법 3 을 비교하기 위해 n = 10,240 인 비트 스트림에 대해 d = n/4 = 2560 인 테스트를 하였다. 표 6 과 표 7 을 보면 두 방법의 power 가 비슷함을 알 수 있다. 또 이 경우 n = 20,480 이고 d = n/2 = 10,240 인 앞의 실험에 비해 H_2 을 따르는 비트 스트림을 어느 정도 검정할 수 있음을 알 수 있다. 그러나, runs test 나 runs distribution test 에 비해서는 power 가 작다.

작은 d 의 값에 대해 비트 스트림을 검정하는 정도를 알아보기 위해 d = 2, 4, 8, 1024 에 대해 방법 3 을 사용하여 H_1 을 따르는 비트 스트림과 H_2 을 따르는 비트 스트림을 검정하였다. 그 결과, 표 6 과 표 7 에서 알 수 있듯이 거리 d 의 변화에 대해 power function 에 대해서는 큰 차이가 없음을 알 수 있었다. 방법 4 의 경우에는 길이 n 인 비트 스트림에 대해 $1 \le d \le n$ 인 모든 d 에 대해 테스트를 하기 때문에 n 이 클 경우 power function 을 구하기가 어렵다. 따라서, 본 논문에서는 실험을 하지 않았다.

비트 스트림을 일정한 길이의 subblock 단위로 나누어서 테스트하는 경우인 § 3.1 의 frequency test 를 subblock 에 적용하는 방법과 Poker test 와 Universal test 의 power function 을 표 8~11 에 나타내었다. 비교 결과 Universal test 의 power 가 다른 테스트보다 작다.

표 12~13 의 결과를 보면 § 3.8 의 binary derivative test 의 경우 2-serial test 와 1-binary derivative test 는 비슷한 power 를 가지고 있다고 생각할 수 있다. 또한 runs test 보다 H_1 을 따르는 비트 스트림을 binary derivative test 가 더 잘 검정을 한다. 그리고, 1-binary derivative test 와 2-binary derivative test 는 비슷한 power 를 가지고 있다.

표 14 는 § 3.10 의 sequence complexity test 를 § 3.10 에서 설명한 threshold 값에 대해 power function 을 구한 결과이다. 표 14 의 sequence complexity test 의 power 는 낮지만 이 경우에는 유의수준 α대신 § 3.10 의 threshold 값을 사용함으로 이 테스트를 통과하지 못하는 비트 스트림은 매우 nonrandom 하다고 할 수 있다.

4.2 스트림 암호 알고리즘의 통계적 특성분석을 위해 효과적인 randomness test 방법

지금까지의 내용을 정리하여 여러 가지 randomness test 방법들 중 다음과 같은 테스트 방법들을 스트림 암호 알고리즘의 통계적 특성 분석을 위하여 효율적인 randomness test 방법으로 제시한다.

Frequency test – § 3.1
Frequency test 를 subblock 에 적용하는 방법 – § 3.1
Runs distribution test 방법 1 – § 3.2
Change point test – § 3.3

Universal test – § 3.6
1-Binary derivative test – § 3.8
Linear complexity test – § 3.10
Sequence complexity test – § 3.11

비트 스트림의 균일성을 검정하기 위하여 가장 보편적이고 널리 알려진 frequency test 를 사용할 것을 제안하고, 비트 스트림을 subblock 으로 나누어서 테스트하는 방법으로 frequency test 방법 3 과 universal test 를 사용할 것을 제안한다. 비록 universal test 의 power 가 다른 테스트에 비해 작지만 subblock 의 entropy 와 관련하여 테스트하는 universal test 가 frequency test 와 동일하다고는 생각할 수 없기 때문이다. Poker test 의 경우도 비트 스트림을 subblock 으로 나누어서 테스트하는 frequency test 와 비슷한 power 를 가졌다고 할 수 있지만 frequency test 를 subblock 에 적용하는 방법을 사용함으로써 poker test 가 테스트할 수 있는 내용을 충분히 포함할 수 있다.

그리고, runs distribution test 의 경우에는 이 테스트를 통과하지 못함은 비트 스트림 내에 특정 길이의 block 또는 gap 들이 상대적으로 다른 길이에 비해 많음을 나타내므로 이는 다른 테스트에서 검사할 수 없다. 또한 본 논문에서 실험을 하지 않았지만 다른 테스트를 통과한 비트 스트림이 포함할 수 있는 nonrandom 한 비트 스트림을 검정할 수 있는[7] change point test 를 사용할 것을 제안한다.

1-binary derivative test 의 경우는 runs test 와 serial test 의 내용을 포함한다고 할 수 있다. 그리고, 스트림 암호 알고리즘의 안전성을 검사하기 위해 꼭 필요한 Linear complexity test 와 비트 스트림내의 동일한 패턴을 가진 subblock 이 너무 많이 존재하는 가를 검사하기 위한 sequence complexity test 를 사용할 것을 제안한다. 이 sequence complexity test 는 또한 비트 스트림의 주기를 검정하기 위한 autocorrelation test 의 방법보다 효율적이다[7].

4.3 블록 암호 알고리즘의 통계적 특성분석을 위하여 효과적인 randomness test 방법

블록 암호 알고리즘은 스트림 암호 알고리즘과 같이 키 스트림의 randomness 를 테스트할 필요가 없으므로 § 3 에서 설명한 randomness test 방법을 그대로 적용할 수가 없다. 따라서, 블록 암호 알고리즘의 평문과 암호문 사이의 독립성을 검정하기 위한 다음과 같은 방법에 randomness test 방법을 사용한다.

방법 1 [4]
Nonrandom 한 비트 스트림을 알고리즘에 평문으로 입력을 할 경우 만일 평문과 암호문이 독립성을 가지고 있다면 암호문은 random 한 비트 스트림이 될 것이다. 따라서, 이 암호문에 대해 randomness test 를 수행한다.

방법 2 [4]
Random 한 비트 스트림을 알고리즘에 평문으로 입력할 경우 생성되는 암호문과 평문을 bitwise XOR 한 결과는 만일 평문과 암호문이 독립성을 가지고 있다면 random 한 비트 스트림이 될 것이다. 이 비트 스트림에 대해 randomness test 를 한다.

평문(또는 키)과 암호문 사이의 독립성을 검정하기 위한 위의 방법 1 과 방법 2 에서 사용하기 위하여 다음과 같은 randomness test 방법을 사용할 것을 제안한다.

Frequency test – § 3.1

Runs distribution test 방법 1 – § 3.2
Frequency test 를 subblock 에 적용하는 방법 – § 3.1 또는 Poker test – § 3.5
Change point test – § 3.3
1-Binary derivative test – § 3.8
Linear complexity test – § 3.10
Sequence complexity test – § 3.11

§ 4.2 에서 스트림 암호 알고리즘을 위해 제시한 방법들과 대부분 일치한다고 할 수 있다. 그러나, 비트 스트림을 subblock 으로 나누어서 테스트하는 방법의 경우, subblock 의 크기가 커지게 되면 chi-square 테스트를 하기 위해 필요한 기대값 e_i가 작아지게 되므로 Frequency test 를 subblock 에 적용하는 방법 대신 poker test 를 사용하면 더 큰 subblock 에 대해서도 테스트가 가능하다. Universal test 는 테스트를 하기 위해 필요한 비트 스트림의 길이가 블록 암호 알고리즘의 블록 크기보다 크므로 사용할 수가 없다.

지금까지는 평문(또는 키)과 암호문의 독립성을 검정하기 위한 방법을 소개하였다. 블록 암호 알고리즘에서는 이 방법 외에 블록 암호 알고리즘이 만족해야 하는 통계적 특성 중 하나인 avalanche 성질을 만족하는 가를 실험하는 것이 중요하다. 따라서, § 3.12 에 설명된 avalanche test 를 블록 암호 알고리즘의 통계적 특성 분석을 위해 사용할 것을 제안한다. Avalanche test 는 평문의 한 비트가 바뀌었을 때 암호문의 각 비트들이 random 하게 바뀌는 가를 측정하는 Plaintext avalanche test 와 키의 한 비트가 바뀌었을 때 암호문의 각 비트들이 random 하게 바뀌는 가를 측정하는 Key avalanche test 를 생각할 수 있다.

Plaintext(Key) avalanche test – § 3.12

5. DES, AES 후보알고리즘, SEED 의 통계적 특성 분석

DES, AES 후보알고리즘, SEED 에 대한 통계적 특성을 분석하여 표 15~18 에 나타내었다. 실험 결과 DES, AES 후보알고리즘, SEED 는 모두 평문과 암호문의 독립성을 검정하는 테스트들에 대해 randomness 를 만족하였으며, avalanche test 또한 만족하였다.

SEED 의 경우, 라운드별로 통계적 특성을 다시 분석하여 표 18~21 에 나타내었다. 실험 결과 2 라운드이후부터는 평문과 암호문의 독립성을 검정하는 테스트들에 대해 randomness 를 만족하였으며, Avalanche test 의 경우에는 4 라운드이후에는 만족하였다.

6.결론

본 논문에서는 지금까지 널리 알려진 여러 가지 randomness test 방법들을 비밀키 암호 시스템의 통계적 분석을 효율적으로 하기 위하여 비교 분석하였다. 대부분의 randomness test 방법들은 비트 스트림들이 독립적이고 동일한(independent and identically distributed) 분포를 가지며 각 비트들의 값이 0 또는 1 이 될 확률이 같다는 영 가정을 근거로 하여 비트 스트림의 randomness 를 측정한다. 따라서, 이러한 영 가정을 벗어나는 비트 스트림에 대해 각각의 randomness test 들이 얼마나 효율적으로 검정을 할 수 있는 가를 측정하기 위하여 power function 의 개념을 사용하였다. 각각의 randomness test 방법들의 power function 을 이용하여 어느 방법이 더욱 효율적인 가를 비교한 후 스트림 암호 알고리즘과 블록 암호 알고리즘의 통계적 특성을 측정하기 위해 가장 효율적이라고 생각되는 테스트 방법들을

찾아 이 방법들을 사용할 것을 제안하였다

끝으로, 본 논문에서 블록 암호 알고리즘의 통계적 특성 분석을 위해 사용하도록 제안한 randomness test 방법들을 이용하여 DES, AES 후보알고리즘, SEED 의 통계적 특성을 분석하였다.

참고문헌

[1] 최 봉대, 신 양우, 한 동환, 최 두일, "Randomness 특성 분석에 관한 연구", *한국전자통신연구소 데이터 보호의 기반 기술 연구 보고서*, 한국 과학 기술원, 12/92.

[2] G. Carter, "Aspects of Local Linear Complexity", *PhD Thesis*, University of London, 1989

[3] P.L. Ecuyer, "Efficient and Portable Combined Random Number Generators", *communications of the ACM*, 31, No.6, R.E.Nance ed.,1988,pp742-774

[4] H. Gustafson, E. Dawson, and B. Caelli, "Comparison of Block Ciphers", *Advances in Cryptology-AUSCRYPT'90*, pp208-220, Spring-Verlag, 1990

[5] H. Gustafson, E. Dawson, L. Nielson and W. Caelli, "Computer Package for Measuring the Strength of Encryption Algorithms"

[6] I.J. Good, " The Serial Test for Sampling Numbers and Other Tests for Randomness", *Processings of the Cambridge Philosophical Society*, 49, 1953, pp. 267-284.

[7] H. Gustafson, "Statistical analysis of symmetric ciphers", *A thesis for the degree of Doctor of Philosophy*, Queensland University, 1996

[8] A. Lempel and J. Ziv, "On the Complexity of Finite Sequences", *IEEE Trans. on Information Theory* Vol. IT-22, Jan. 1976, pp 75-81.

[9] A.K. Leung and S.E. Tavares. "Sequence complexity as a test for cryptographic systems", *In Advances in Cryptology: CRYPTO'84*, pp 468-474, Springer-Verlag, 1985

[10] J.L. Massey, "Shift Register Sequences and BGH Decoding", *IEEE Trans. on Information Theory*, Vol. IT-15, Jan. 1969, pp. 122-127

[11] U.M. Maurer, "A Universal Statistical Test for Random Bit Generators", *Journal of Cryptology*, 1992, 5, pp.89-105

[12] A.M. Mood, "The Distribution Theory of Runs", *Annals of Mathematical Statistics*, Vol 11, 1940, pp. 367-392

[13] A.G. Konheim, *Cryptography* – a Primer, Wiley, New York, 1981.

[14] A.N. Pettit, "A Non-parametric Approach to the Change-point Problem", *Applied Statistics*, 28 No. 2, 1979, pp. 126-135

[15] R.A. Rueppel, "Analysis and Design of Stream Ciphers", Springer-Verlag, 1986

[16] D. Wagner, "The Security of MacGuffin", *A thesis for the degree of bachelor of Arts in the department of mathematics*, Princeton University, 1995

[17] F.M. Willems, "Universal data compression and repetition times", *IEEE Transactions on Information Theory*, Vol. 35, Jan. 1989, pp.54-58

[18] J. Ziv and A. Lempel, "A Universal algorithm for sequential data compression", *IEEE Transactions on Information Theory*, vol. 23, May 1977, pp.337-343

[19] http://www.kisa.or.kr/ 1998/11/25

[20] J.M. Carroll and L.E. Robbins, "Using Binary Derivatives to Test an enchancement DES", *Cryptologia*, Vol. XII Number 4, Oct. 1988, pp.193-208

[21] National Bureau of Standards, "Data Encryption Standard", U. S. Department of Commerce, *NBS FIPS PUB 46*, Jan. 1977.

[22] National Institute of Standards and Technology, "Announcing the Data Encryption Standard DES", *FIPS PUB 46-2*, 1993

[23] National Institute of Standards and Technology, "Draft AES Announcement", http://csrc.nist.gov/encryption/aes/aes_home.htm.

p or 1-p	Power Function									
	α=0.05					α=0.01				
	Freq	Runs	Runs distribution			Freq	Runs	Runs distribution		
			방법 1	방법 2	방법 3			방법 1	방법 2	방법 3
0.50	0.0526	0.0456	0.0625	0.0333	0.1265	0.0102	0.0084	0.0148	0.0077	0.0886
0.49	0.5199	0.0472	0.1991	0.1233	0.2339	0.2880	0.0087	0.0670	0.0395	0.1593
0.48	0.9812	0.0495	0.7207	0.6265	0.5325	0.9236	0.0095	0.5043	0.3960	0.3693
0.47	1.0000	0.0648	0.9902	0.9805	0.8783	0.9998	0.0133	0.9579	0.9264	0.7552
0.46	1.0000	0.0948	1.0000	0.9999	0.9924	1.0000	0.0249	0.9998	0.9993	0.9740
0.45	1.0000	0.1699	1.0000	1.0000	0.9999	1.0000	0.0569	1.0000	1.0000	0.9995
0.44	1.0000	0.3073	1.0000	1.0000	1.0000	1.0000	0.1297	1.0000	1.0000	1.0000
0.42	1.0000	0.7362	1.0000	1.0000	1.0000	1.0000	0.5067	1.0000	1.0000	1.0000
0.40	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

표 2 : Power function (H1) : Frequency, Runs, Runs distribution 방법 1, 2, 3 n = 10240

q or 1-q	Power Function									
	α=0.05					α=0.01				
	Freq	Runs	Runs distribution			Freq	Runs	Runs distrituion		
			방법 1	방법 2	방법 3			방법 1	방법 2	방법 3
0.50	0.0487	0.0537	0.0658	0.0348	0.1323	0.0093	0.0098	0.0155	0.0080	0.0908
0.49	0.0562	0.5237	0.3233	0.2377	0.2498	0.0143	0.2843	0.1601	0.1075	0.1656
0.48	0.0625	0.9815	0.8878	0.8314	0.6102	0.0146	0.9223	0.7560	0.6754	0.4416
0.47	0.0669	1.0000	0.9984	0.9974	0.9419	0.0154	0.9998	0.9949	0.9905	0.8652
0.46	0.0692	1.0000	1.0000	1.0000	0.9989	0.0180	1.0000	1.0000	1.0000	0.9931
0.45	0.0755	1.0000	1.0000	1.0000	1.0000	0.0215	1.0000	1.0000	1.0000	1.0000
0.43	0.0821	1.0000	1.0000	1.0000	1.0000	0.0242	1.0000	1.0000	1.0000	1.0000
0.41	0.0962	1.0000	1.0000	1.0000	1.0000	0.0280	1.0000	1.0000	1.0000	1.0000
0.39	0.1176	1.0000	1.0000	1.0000	1.0000	0.0425	1.0000	1.0000	1.0000	1.0000
0.37	0.1271	1.0000	1.0000	1.0000	1.0000	0.0456	1.0000	1.0000	1.0000	1.0000
0.35	0.1529	1.0000	1.0000	1.0000	1.0000	0.0595	1.0000	1.0000	1.0000	1.0000
0.33	0.1610	1.0000	1.0000	1.0000	1.0000	0.0720	1.0000	1.0000	1.0000	1.0000
0.31	0.1877	1.0000	1.0000	1.0000	1.0000	0.0853	1.0000	1.0000	1.0000	1.0000

표 3 : power function (H2) : : Frequency, Runs, Runs distribution 방법 1, 2, 3 n = 10240

p or 1-p	Power Function					
	α=0.05			α=0.01		
	방법 1	방법 2	방법 3	방법 1	방법 2	방법 3
0.50	0.047000	0.131400	0.047000	0.009900	0.027900	0.009900
0.49	0.045400	0.612900	0.045400	0.009600	0.330900	0.009600
0.48	0.051200	0.988600	0.051200	0.010900	0.933600	0.010900
0.47	0.061200	1.000000	0.061200	0.014100	0.999700	0.014100
0.46	0.094400	1.000000	0.094400	0.024600	1.000000	0.024600
0.45	0.170700	1.000000	0.170700	0.053200	1.000000	0.053200
0.43	0.509500	1.000000	0.509500	0.279100	1.000000	0.279100
0.41	0.905300	1.000000	0.905300	0.756900	1.000000	0.756900
0.39	0.999200	1.000000	0.999200	0.990800	1.000000	0.990800
0.37	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

표 4 : Power function (H1) : Autocorrelation test, n = 20480, d=10240

q or 1-q	Power Function					
	α=0.05			α=0.01		
	방법 1	방법 2	방법 3	방법 1	방법 2	방법 3
0.50	0.047600	0.135700	0.047600	0.009800	0.027000	0.009800
0.48	0.049500	0.149700	0.049500	0.009700	0.032900	0.009700
0.46	0.054200	0.161700	0.054200	0.012300	0.039700	0.012300
0.44	0.053600	0.182000	0.053600	0.010600	0.045900	0.010600
0.42	0.052700	0.194100	0.052700	0.012100	0.051900	0.012100
0.40	0.058300	0.210700	0.058300	0.012300	0.061900	0.012300
0.38	0.065400	0.233500	0.065400	0.017300	0.076800	0.017300
0.36	0.066000	0.258700	0.066000	0.013800	0.084500	0.013800
0.34	0.074900	0.276600	0.074900	0.018400	0.105700	0.018400
0.32	0.082600	0.310300	0.082600	0.022100	0.123200	0.022100

표 5 : Power function (H2) : Autocorrelation test, n=20480, d = 10240

p or 1-p	Power Function					
	α=0.05					
	d = 2560		방법 3			
	방법 1	방법 3	d = 2	d = 4	d=8	d = 1024
0.50	0.047100	0.053800	0.048500	0.049900	0.048000	0.049000
0.48	0.049900	0.053900	0.050900	0.054100	0.051700	0.052300
0.46	0.072900	0.091800	0.099400	0.102300	0.093900	0.093600
0.44	0.173100	0.246300	0.304100	0.313200	0.296200	0.278500
0.42	0.442200	0.604100	0.729200	0.736600	0.725400	0.687200
0.40	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

표 6 : Power function (H1) : Autocorrelation test, n = 10240

q or 1-q	Power Function					
	α=0.05					
	d = 2560		방법 3			
	방법 1	방법 3	d = 2	d = 4	d=8	d = 1024
0.50	0.051900	0.050000	0.044600	0.050800	0.049900	0.049600
0.48	0.049400	0.051900	0.048500	0.054900	0.049900	0.049600
0.46	0.052400	0.052900	0.093400	0.052400	0.049200	0.047100
0.44	0.052200	0.054400	0.303600	0.052100	0.051000	0.054300
0.42	0.052600	0.055600	0.733200	0.057300	0.058300	0.051300
0.40	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

표 7 : Power function (H2) : Autocorrelation test, n = 10240

p or 1-p	Power Function					
	α=0.05			α=0.01		
	Universal	Freq	Poker	Universal	Freq	Poker
0.50	0.079000	0.051000	0.046000	0.015000	0.010000	0.016000
0.48	0.176000	1.000000	1.000000	0.069000	1.000000	1.000000
0.46	0.879000	1.000000	1.000000	0.734000	1.000000	1.000000
0.44	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

표 8 : Power function (H1) : Universal test, b=4, n=102400

p or 1-p	Power Function					
	α=0.05			α=0.01		
	Universal	Freq	Poker	Universal	Freq	Poker
0.50	0.059000	0.033000	0.036000	0.019000	0.006000	0.008000
0.49	0.090000	1.000000	1.000000	0.024000	1.000000	1.000000
0.48	0.428000	1.000000	1.000000	0.206000	1.000000	1.000000
0.47	0.954000	1.000000	1.000000	0.894000	1.000000	1.000000
0.46	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

표 9 : Power function (H1) : Universal test, b=6, n=387840

q or 1-q	Power Function					
	α=0.05			α=0.01		
	Universal	Freq	Poker	Universal	Freq	Poker
0.50	0.082000	0.055000	0.045000	0.023000	0.001000	0.013000
0.49	0.086000	0.978000	0.913000	0.024000	0.738000	0.764000
0.48	0.132000	1.000000	1.000000	0.052000	1.000000	1.000000
0.47	0.308000	1.000000	1.000000	0.149000	1.000000	1.000000
0.45	0.957000	1.000000	1.000000	0.880000	1.000000	1.000000
0.43	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

표 10 : Power function (H2) : Universal test, b=4, n=102400

q or 1-q	Power Function					
	α=0.05			α=0.01		
	Universal	Freq	Poker	Universal	Freq	Poker
0.50	0.059000	0.046000	0.055000	0.010000	0.005000	0.016000
0.49	0.080000	1.000000	1.000000	0.019000	1.000000	0.999000
0.47	0.875000	1.000000	1.000000	0.728000	1.000000	1.000000
0.45	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

표 11 : Power function (H2) : Universal test, b=6, n=387840

p or 1-p	Power Function							
	α=0.05				α=0.01			
	1-BD	2-BD	2-Serial	Runs	1-BD	2-BD	2-Serial	Runs
0.50	0.048500	0.047400	0.049100	0.045600	0.008600	0.009300	0.010100	0.008400
0.49	0.421400	0.375900	0.415500	0.047200	0.215800	0.184200	0.205400	0.008700
0.48	0.962500	0.950700	0.960400	0.049500	0.883600	0.861400	0.868500	0.009500
0.47	1.000000	0.999800	1.000000	0.064800	0.999400	0.998900	0.999200	0.013300
0.46	1.000000	1.000000	1.000000	0.094800	1.000000	1.000000	1.000000	0.024900
0.44	1.000000	1.000000	1.000000	0.307300	1.000000	1.000000	1.000000	0.129700
0.42	1.000000	1.000000	1.000000	0.736200	1.000000	1.000000	1.000000	0.506700
0.40	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

표 12 : Power function (H1) : 1-BD, 2-BD, Serial, Runs, n = 10240

q or 1-q	Power Function							
	α=0.05				α=0.01			
	1-BD	2-BD	2-Serial	Runs	1-BD	2-BD	2-Serial	Runs
0.50	0.051200	0.047900	0.050100	0.053700	0.008700	0.008900	0.009000	0.009800
0.49	0.430300	0.378700	0.425700	0.523700	0.220300	0.187900	0.213800	0.284300
0.48	0.963700	0.950100	0.959100	0.981500	0.886100	0.863400	0.870100	0.922300
0.47	1.000000	0.999800	0.999900	1.000000	0.999400	0.998900	0.998900	0.999800
0.46	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

표 13 : Power function (H2) : 1-BD, 2-BD, Serial, Runs, n = 10240

p or 1-p	Power Function	q or 1-q	Power Function
	threshold = 768.657504		threshold = 768.657504
0.50	0.000000	0.50	0.000000
0.48	0.000000	0.48	0.000000
0.46	0.000000	0.46	0.000000
0.44	0.000500	0.44	0.000500
0.42	0.076400	0.42	0.044600
0.40	1.727200	0.40	1.000000
0.38	0.998600	0.38	1.000000
0.37	1.000000	0.37	1.000000

표 14 : power function (H1, H2) : Sequence complexity test, n = 10240

알고리즘	통계적 특성											
	Frequecy test				Runs distribution test				Frequency test on subblock(b=2)			
	% of failed samples				% of failed samples				% of failed samples			
	유의수준				유의수준				유의수준			
	5%		1%		5%		1%		5%		1%	
	방법		방법		방법		방법		방법		방법	
	1	2	1	2	1	2	1	2	1	2	1	2
DES	5.45	6.26	0.82	0.91	8.14	7.81	1.58	1.82	4.75	4.67	0.87	0.86
CAST256	4.32	4.13	1.01	1.03	6.96	7.08	1.77	1.71	4.94	5.07	0.98	0.93
CRYPTON	4.31	3.89	1.07	0.84	7.37	6.67	1.90	1.52	5.17	4.85	1.02	0.80
DEAL	4.16	4.15	0.91	1.09	6.99	7.04	1.64	1.65	4.91	4.68	0.97	0.83
DFC	4.23	3.85	1.09	0.90	7.35	7.23	1.87	1.92	4.88	4.96	1.00	0.97
E2	4.23	4.02	1.13	0.87	7.32	7.10	1.71	1.52	5.15	4.86	1.00	1.04
FROG	4.46	3.89	1.06	0.92	7.44	7.40	1.73	1.92	5.01	4.57	1.06	0.84
HPC	4.21	3.50	0.88	0.81	7.31	6.93	1.81	1.68	5.12	4.68	0.93	0.87
LOKI97	4.31	4.03	1.02	0.89	7.38	6.77	1.71	1.87	0.50	5.10	1.03	1.01
MAGENTA	4.21	4.41	1.08	1.19	6.99	7.44	1.69	2.17	5.07	4.98	1.05	0.97
MARS	4.01	4.44	1.02	1.01	7.31	7.50	1.87	1.77	4.61	5.14	0.83	0.93

RC6	4.23	3.94	0.98	0.98	7.43	7.27	1.69	1.73	4.94	4.77	0.87	0.90
RIJNDAEL	4.00	4.25	0.96	1.03	7.39	6.92	1.82	1.83	4.81	4.97	0.90	0.91
SAFER	4.11	3.69	0.93	0.91	7.26	7.17	1.68	1.90	5.09	4.85	0.85	0.99
SERPENT	4.23	4.15	1.07	0.91	7.26	7.34	1.84	1.83	4.96	5.27	1.02	1.06
TWOFISH	4.17	4.24	0.96	0.74	7.27	7.24	1.82	1.85	4.99	5.22	0.93	1.00
SEED	3.91	4.53	0.96	1.26	7.20	7.81	1.65	2.08	4.96	5.61	0.91	1.29

표 15: 알고리즘의 통계적 특성분석 I (평문과 암호문 사이의 독립성, 방법 1/방법 2)

알고리즘	통계적 특성											
	Poker test(b=4)				Change point test				1-Binary derivative test			
	% of failed samples				% of failed samples				% of failed samples			
	유의수준				유의수준				유의수준			
	5%		1%		5%		1%		5%		1%	
	방법		방법		방법		방법		방법		방법	
	1	2	1	2	1	2	1	2	1	2	1	2
DES	5.21	5.34	1.32	1.08	3.27	3.51	0.38	0.62	5.02	5.19	0.87	0.81
CAST256	5.41	5.15	1.28	1.32	3.94	3.96	0.82	0.80	4.59	4.53	0.93	0.99
CRYPTON	5.18	4.83	1.31	1.11	3.85	3.90	0.87	0.78	4.58	4.23	1.02	0.89
DEAL	5.34	5.22	1.30	1.21	3.89	3.73	0.78	0.74	4.50	4.36	0.93	1.10
DFC	4.89	4.94	1.21	1.25	4.17	4.16	0.85	0.73	4.73	4.67	1.15	0.98
E2	5.11	4.81	1.16	1.30	4.19	4.07	0.70	0.73	4.73	4.44	1.02	0.95
FROG	5.27	4.98	1.45	1.43	4.06	3.72	0.74	0.70	4.75	4.49	1.10	0.96
HPC	5.34	4.53	1.29	1.20	3.67	3.87	0.64	0.67	4.77	4.46	1.08	0.88
LOKI97	5.27	5.07	1.38	1.22	4.16	4.19	0.85	0.73	4.97	4.70	0.96	0.90
MAGENTA	5.21	5.31	1.20	1.43	3.75	3.89	0.70	0.80	4.79	5.27	1.11	1.15
MARS	4.93	5.46	1.24	1.39	3.93	3.96	0.65	0.70	4.51	4.92	0.93	1.03
RC6	5.20	4.99	1.22	1.26	3.98	4.19	0.66	0.69	4.63	4.56	0.96	0.90
RIJNDAEL	5.06	4.91	1.31	1.30	3.94	3.96	0.61	0.75	4.64	4.71	0.94	0.97
SAFER	4.95	4.58	1.19	1.10	4.19	3.84	0.73	0.72	4.61	4.44	0.91	1.06
SERPENT	5.14	4.92	1.33	1.39	3.92	4.00	0.66	0.81	4.52	4.70	1.07	1.04
TWOFISH	5.38	5.38	1.33	1.36	3.83	5.52	0.69	0.49	4.65	4.42	1.03	0.85
SEED	4.94	5.36	1.25	1.35	3.97	4.03	0.87	0.79	4.52	5.25	1.02	1.25

표 16: 알고리즘의 통계적 특성분석 II (평문과 암호문 사이의 독립성, 방법 1/방법 2)

알고리즘	통계적 특성			
	Linear complexity test		Sequence complexity test	
	α = 0.05, 자유도 10 threshold value = 18.3070		% of samples less than threshold value	
	방법 1	방법 2	방법 1	방법 2
DES	10.76(자유도 9/th=16.9191)	12.9916(자유도 9/th=16.9191)	0.07	0.10
CAST256	6.5354	10.9564	0.18	0.21
CRYPTON	17.3048	9.7844	0.24	0.20
DEAL	8.0662	14.1982	0.18	0.13
DFC	17.0311	9.7086	0.19	0.13
E2	9.0728	7.0220	0.18	0.19
FROG	10.8202	6.0724	0.18	0.16
HPC	8.9636	8.0204	0.15	0.23
LOKI97	4.5556	14.4136	0.16	0.15
MAGENTA	10.6714	7.2682	0.16	0.16
MARS	4.4014	10.8198	0.16	0.25
RC6	10.1944	8.485	0.13	0.16
RIJNDAEL	7.4654	12.1396	0.21	0.24
SAFER	13.4940	12.4020	0.16	0.24
SERPENT	10.6818	12.2960	0.22	0.14
TWOFISH	11.2594	8.5838	0.17	0.15
SEED	18.1484	10.0808	0.19	0.17

표 17 : 알고리즘의 통계적 특성분석 III (평문과 암호문 사이의 독립성, 방법 1/방법 2)

알고리즘	Avalanche test	SEED 의 라운드별 Avalanche test	
	$63.9933 \leq X \leq 64.0067$		
CAST256	63.9991	라운드수	$63.9933 \leq X \leq 64.0067$
CRYPTON	64.0006	2	63.3396
DEAL	64.0030	4	63.9987
DFC	64.0002	6	64.0059
E2	64.0021	8	63.9995
FROG	63.9994	10	64.0014
HPC	63.9988	12	64.0011
LOKI97	63.9989	14	63.9993
MAGENTA	64.0067	16	63.9957
MARS	63.9963		
RC6	64.0031		
RIJNDAEL	63.9995		
SAFER	64.0016		
SERPENT	64.0030		
TWOFISH	63.9977		
SEED	63.9957		

표 18 : 알고리즘의 통계적 특성분석 IV(Avalanche test)

라운드 수		Frequency test		Runs distribution test		Frequency test on subblock(b=2)	
		% of failed samples		% of failed samples		% of failed samples	
		유의수준		유의수준		유의수준	
		5%	1%	5%	1%	5%	1%
방법 1	2	2.82	0.58	7.92	1.84	3.03	0.45
	4	4.26	1.02	7.15	1.68	5.15	0.99
	6	4.29	1.20	7.40	1.73	5.15	1.02
	8	4.05	1.06	7.16	1.71	4.92	1.05
	10	4.13	0.92	7.13	1.82	4.94	1.02
	12	4.06	0.90	7.57	1.82	4.94	1.02
	14	4.12	1.03	7.10	1.68	5.09	1.01
	16	3.91	0.96	7.20	1.65	4.96	0.91

방법 2	2	4.30	0.92	7.28	1.64	4.99	0.89
	4	4.33	1.03	7.29	1.88	4.93	1.10
	6	5.93	1.30	7.28	1.54	4.95	0.92
	8	4.10	0.86	7.72	1.93	4.89	0.94
	10	4.03	0.99	7.39	1.86	5.09	0.87
	12	4.20	1.20	7.52	1.87	5.03	0.96
	14	4.17	1.05	7.30	1.82	5.31	1.11
	16	4.53	1.26	7.81	2.08	5.61	1.29

표 19 : SEED 의 통계적 특성분석 (평문과 암호문 사이의 독립성, 방법 1/방법 2)

라운드 수		Poker test(b=4)		Change point test		1-Binary derivative test	
		% of failed samples		% of failed samples		% of failed samples	
		유의수준		유의수준		유의수준	
		5%	1%	5%	1%	5%	1%
방법 1	2	3.43	0.82	3.54	0.64	3.42	0.63
	4	5.14	1.28	4.24	0.82	4.61	0.89
	6	5.13	1.23	4.02	0.77	4.90	1.15
	8	5.01	1.18	3.72	0.65	4.62	1.05
	10	5.12	1.07	4.26	0.78	4.44	1.02
	12	5.06	1.30	4.03	0.64	4.55	0.90
	14	5.19	1.45	3.95	0.65	4.60	1.02
	16	4.94	1.25	3.97	0.87	4.52	1.02
방법 2	2	5.60	0.15	3.87	0.73	4.51	0.89
	4	5.47	1.37	4.00	0.64	4.93	1.18
	6	5.09	1.31	4.02	0.67	4.70	1.06
	8	4.88	1.28	3.91	0.69	4.66	0.98
	10	5.14	1.26	3.84	0.79	4.69	1.00
	12	4.83	1.22	4.23	0.82	4.85	1.09
	14	5.27	1.33	3.91	0.82	4.67	1.10
	16	5.36	1.35	4.03	0.79	5.25	1.25

표 20 : SEED 의 통계적 특성분석 II (평문과 암호문 사이의 독립성, 방법 1/방법 2)