이중 인벌루션 구조를 지니는 가변길이 블록 암호 알고리즘

○
이 인 실, 심 경 섭, 김 혜 정, 신 원, 신 상 욱, 이 경 현
부경대학교 전자계산학과

A variable length Block Algorithm with Double Involution - BADI

○
In-Sil Lee, Kyung-Sub Sim, Hae-Jeong Kim, Weon Shin,
Sang-Uk Shin, Kyung-Hyune Rhee
Dept. of Computer Science, Pukyong Nat'l University

요 약

본 논문에서는 새로운 가변길이 블록 암호알고리즘을 제안한다. 제안된 블록 암호알고리즘은 128비트에서 256비트까지의 가변적인 키 길이를 가지며 가변적인 라운드 수를 사용한다. 각 라운드는 서로 다른 두 개의 F함수를 사용하여 2단계로 구성되는 double involution구조를 사용하며, 또한 두 개의 서로 다른 키 스케줄링 알고리즘을 사용하여 알려진 공격에 대해 안전하도록 설계하였다.

1. 서론

많은 양의 메시지를 빠른 처리 속도로 암호화하는데 있어서 블록 암호알고리즘이 적합한데 본 논문에서는 새로운 안전한 블록 암호알고리즘을 제안한다. 기존의 블록 암호알고리즘으로는 DES(Data Encryption Standard), IDEA(International Data Encryption Algorithm), SAFER(Secure And Fast Encryption Routine), MISTY 등이 있는데 DES는 1973년 NBS(National Bureau of Standard)가 암호 시스템을 등록시키기 위해 공고하였고 1977년 표준으로 채택(FIPS(Federal Information Process Standard) 46)하였으며 1997년까지 미국 표준으로 사용하였다. DES는 64비트의 블록크기와 56비트의 키 길이를 가지며 1998년 이후 DES의 대체 알고리즘으로 AES(Advanced Encryption Standard)를 개발추진중이다. IDEA는 1990년 Lai와 Massey가 PES(Proposed Encryption Standard)를 개발하다가 DC(Differential Cryptanalysis)[1]공격에 강하도록 개선한 IPES(Improved PES)를 1991년에 개발하였으며 1992년에 IDEA라는 이름으로 바꾸어서 쓰게 되었다. IDEA는 64비트의 블록크기와 128비트의 키 길이를 가지며 DC공격에 강하도록 키에 의존하는 S-box를 사용하며 PGP의 기밀성 서비스를 제공하는데 사용하고 있다. SAFER는 1993년 Massey가 개발하였으며 64비트의 블록크기와 64비트의 키 길이를 가지며 키 스케줄링을 제외한 모든 부분이 바이트 단위로 구성되어 있어 소프트웨어적으로 구현이 용이하므로 스마트 카드용으로 적합하다[2][3]. MISTY는 Matsui, Ichigawa, Sorimachi, Tokita, Yanagishi의 첫 자를 따서 만든 알고리즘으로 64비트의 블록크기와 128비트의 키 길이를 가지며 Feistel 구조에서 라운드 함수 내부를 한번 더 Feistel 구조로 설계하였고 S-box의 크기를 다르게 사용하므로 대수적 공격에 강하다[4][5].

본 논문에서 제안하는 블록 암호알고리즘은 128비트의 블록크기, 가변적인 키 길이와 두 개의 키 스케줄링을 가지며 라운드도 가변적인 라운드를 사용할

수 있게끔 알고리즘의 용도에 따라서 다양한 가변성을 부가하였다. 이러한 알고리즘은 멀티미디어 데이터 암호화에 QoS(Quality of Service)에 따르는 각 서비스별 암호화에 개별적으로 적용하는데 유용하게 이용할 수 있다. 또한 라운드마다 두 단계로 나누어서 서로 다른 두 개의 F함수를 적용한 이중 인벌루션(double involution)구조를 가짐으로써 알려진 공격에 안전하도록 설계하였다.

2. 전체 구조 및 구조도

2.1 기본구조

본 논문에서 제안하는 블록 암호 알고리즘은 Feistel 구조를 기본 구조로 한다. Feistel 구조는 암호화와 복호화가 같은 알고리즘으로 복호화시에는 서브키만 역순으로 적용하는 구조로 라운드에 상관없이 역변환이 가능하고 알고리즘 수행 속도가 빠르다. 또한, 하드웨어, 소프트웨어적으로 구현이 용이하다는 장점이 있다. 본 논문에서 제안한 블록 암호 알고리즘은 이중 인벌루션 구조로 되어 있으며 구성에 있어서 confusion과 diffusion 효과가 최소 라운드 내에서 이루어지도록 구성하였고 두 개의 키 스케줄링으로 구성하였으며 F함수로는 두 개의 함수를 사용한다.

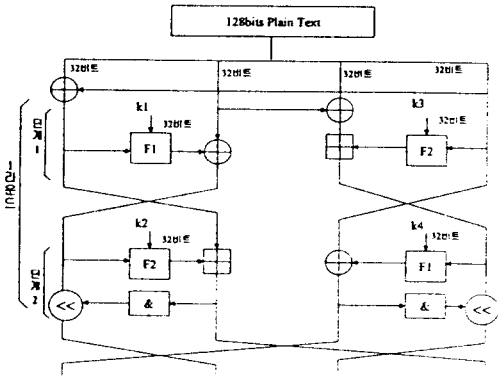
기본적인 라운드의 수는 8라운드로 설계하며 복잡도를 높이기 위해 4의 배수로 증가시킬 수 있도록 라운드를 가변적으로 설계하고 DC공격의 대상이 되는 S-box대신 DC와 LC(Linear Cryptanalysis)[6]에 강한 부울함수를 사용한다[7]. 부울함수는 비 선형적이고 입력의 한 비트가 차이가 나면 출력은 최소 두 비트가 차이가 나는 SAC(Strict Avalanche Criteria)를 만족하며 출력 비트에 독립적이고 bijection을 만족하도록 설계한다. 키 스케줄링은 각각의 입력의 균등성을 보장하고 있고 모든 연산은 32비트 단위로 수행되므로 수행속도가 빠르다.

2.2 전체적인 구조

제안된 블록 암호 알고리즘은 128비트의 평문 블록을 입력으로 받아 128비트의 암호문 블록을 출력하며 128비트, 160비트, 256비트의 가변적인 키를 사용한다. 초기 입력 평문에 키를 적용시켜 반복할 때 각각의 반복 단계를 라운드라고 하며 라운드의 수는 안전성과 속도를 고려해 볼 때 서로 반비례한다. 따라서 본 논문에서 제안하는 블록 암호 알고리즘은 기본적으로는 8라운드를 제안하며 안전성에 따라 4라운드씩 $8+4j$ $(j=0,1,2,\cdots)$로 라운드 수를 가변적으로 구성하였다. 사용되는 두 개의 F함수는 서로 다른 대수적 군에서 연산을 하는데 mod $2^{16}+1$ 곱셈과 mod 2^{16} 덧셈을 결합하고 암호학적으로 강한 부울 함수를 사용한다. 사용된 연산은 mod 2^{16} 덧셈, mod $2^{16}+1$ 곱셈, 순환쉬프트, 비트 단위 논리 연산, 즉 AND, OR, NOT, XOR을 사용한다.

2.3 암호화

암호화는 각 라운드가 2단계로 구성되며 입력 128비트를 64비트씩 왼쪽과 오른쪽으로 나누고 나눈 64비트를 다시 단계 1과 단계 2에서 각각 32비트씩 왼쪽과 오른쪽으로 나누는 involution Feistel 구조를 가지며 두 단계로 나누어 F함수의 순서를 바꾸어서 적용한다. 전체적인 구조도는 <그림 1>과 같다.



<그림 1>전체 구조도

"⊞"기호는 mod 2^{32} 덧셈이고, "⊕"는 비트별 XOR이며 "<<" 는 왼쪽으로 순환쉬프트이고 "&"는 비트별 AND 연산이다.

마지막 라운드 후에는 변수를 swap하지 않고 바로 128비트 암호문으로 출력한다.

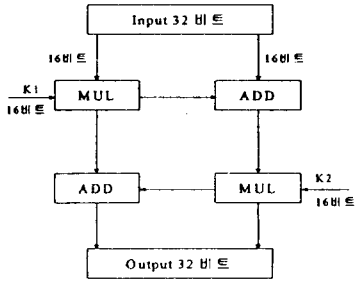
2.4 복호화

복호화는 암호화와 같은 알고리즘을 사용하고 단지 암호화의 역순으로 수행하며 키, 역시 역순으로 적용한다. 연산에 있어서도 mod 2^{16} 덧셈은 mod 2^{16} 뺄셈(2의 보수를 이용한 덧셈)으로 왼쪽 순환쉬프트는 오른쪽 순환쉬프트를 한다.

2.5 F1함수

서로 다른 대수적 군에 의해 연산을 하므로써 confusion 효과를 높일 수 있으며 입력 32비트를 상위비트와 하위비트로 16비트씩 분할해서 수행하고 입력 키 32비트 역시 상위비트와 하위비트 16비트씩

서브키 K1과 K2로 분할한다. MUL은 mod $2^{16}+1$ 곱셈이고 ADD는 mod 2^{16} 덧셈이다. 이 때 입력 키가 64비트로 확장되더라도 32비트씩 서브키로 분할처리가 가능하므로 확장성이 용이하다. 간단하게 구조도로 나타내면 <그림 2>와 같다.



<그림 2> F1함수의 구조도

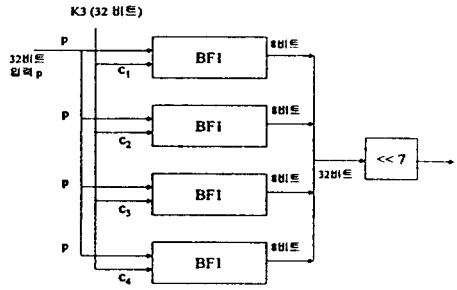
2.6 F2함수

F2함수는 암호학적으로 강한 부울 함수로 구성하는데 먼저, 32비트의 키 K3은 4개의 8비트 (c_1, c_2, c_3, c_4)로 분할되어 각각 부울 함수 BF-1, BF-2, BF-3, BF-4에 입력되고 32비트 입력 P는 4개의 부울 함수에 동일하게 사용된다. 각각의 부울 함수에서는 32비트 입력 P를 8비트 (a, b, c, d)로 나누어서 적용한다. 사용되는 부울 함수는 다음과 같고 <그림 3>에 간단한 구조도를 나타내었다.

$$BF1(a,b,c,d,c_1)$$
$$=(a\&b)\hat{}(c\&d)\hat{}((b\&c\&d)\hat{}c_1)$$
$$BF2(a,b,c,d,c_2)$$
$$=(b\hat{}((d\&c_2)|(a\&c)))$$
$$BF3(a,b,c,d,c_3)$$
$$=(a\hat{}(b\&(a\hat{}d))\hat{}(((a\&d)\hat{}c)\&c_3))$$
$$BF4(a,b,c,d,c_4)$$
$$=((a\&c_4)\hat{}b\hat{}(c\&d))$$

이러한 5변수 부울함수는 다음과 같은 암호학적으로 강한 성질들을 만족한다.

첫 번째 성질은 출력수열의 0과 1이 균형(balanced)을 이루고 두 번째 성질은 SAC을 만족하며 세 번째 성질은 높은 비선형성(non-linearity)을 가지는 것이다.



<그림 3> F2함수의 구조도

3. 키 스케줄링

적은 양의 키 비트를 가지고 많은 양의 라운드 키를 생성하는 것이 키 스케줄링의 목적으로 설계 원칙은 먼저 라운드 키 생성에 있어서 모든 키 비트가 사용되어야 하고 모든 키 비트가 각 라운드마다 독립적으로 새로운 키를 생성해야 한다. 또한, 적어도 one-way를 만족해야 하고 취약키가 없어야 하며 라운드 변환에서의 대칭성과 라운드간의 대칭성을 제거해야 한다. 본 알고리즘에서는 128비트, 160비트, 256비트의 가변길이의 키를 사용하며 키 스케줄링도 이원적인 키 스케줄링(K_1 과 K_2)을 사용한다.

3-1. K_1 스케줄링

K_1 스케줄링의 단계를 살펴보면

1) K_1 을 4개의 32비트로 분할하고 각각을 fk_1, fk_2, fk_3, fk_4 라고 한다.
2) 4개의 32비트 레지스터 R1, R2, R3, R4를 초기화한다.
   R1 = DF7BD629, R2 = E9DB362F, R3 = 5D00F20F, R4 = C3D11FD2
3) i번째 라운드의 서브키중 K_1 스케줄링에서는 k1과 k3을 생성한다.

$$fk_1 = ((fk_1 \wedge R1) \vee (\neg fk_1 \wedge R2))^{\ll R4\&5} \oplus R3$$
$$fk_2 = (fk_2 \oplus R2 \oplus R3) + R4^{\ll R1}$$
$$fk_3 = ((fk_3 \wedge R3) \oplus (fk_3 \wedge R4) \oplus (R3 \wedge R4)) + fk_1$$
$$fk_4 = ((fk_4 \wedge R1) \vee (R4 \wedge \neg R1)) + fk_2$$
$$fk_1 = fk_1 \oplus fk_3 + R2$$
$$fk_2 = fk_2 + fk_4 \oplus R3$$
$$R1 = R3,\ R2 = R4,\ R3 = fk_3^{\ll 11},\ R4 = fk_1$$

생성된 서브키 중에서 fk_1과 fk_2는 라운드 i의 k1과 k3으로 사용한다.

4) i를 1증가시켜서 단계 3)을 반복하여 모든 8라운드에 대한 서브키를 생성한다.

3.2 K_2 스케줄링

K_2 스케줄링의 단계를 살펴보면

1) K_2를 4개의 32비트로 분할하고 각각을 sk_1, sk_2, sk_3, sk_4라고 한다.

2) 두 개의 32비트 레지스터 R1과 R2를 초기화한다.
R1 = 589B4312 , R2 = 91EB718E

3) i번째 라운드의 서브키 k2와 k4를 생성한다.

$$sk_1 = sk_1 \oplus R1^{\lll 7} \oplus R2$$

$$sk_2 = sk_2 \boxplus sk_1 \oplus R2^{\lll 5}$$

$$sk_3 = (sk_3 \oplus sk_2)^{\lll 7}$$

$$sk_4 = sk_4 \boxplus sk_3$$

$$sk_1 = sk_1 \oplus sk_2$$

$$sk_2 = sk_3 \boxplus sk_4$$

$$R1 = sk_4 , R2 = sk_3^{\lll 11}$$

생성된 서브키 중에서 sk_1과 sk_2는 라운드 i의 k2와 k4로 사용한다.

4) i를 1증가시켜서 단계 3)을 반복하여 모든 8라운드에 대한 서브키를 생성한다.

4. 알고리즘의 안전성

기존의 블록 암호 알고리즘에서는 하나의 인벌루션 구조를 사용하는데 반해 제안된 알고리즘의 각 라운드는 서로 다른 두 개의 F함수를 사용하는 이중 인벌루션 구조를 사용하므로서 알려진 공격에 대해 기존의 알고리즘보다 높은 안전성을 제공한다.

F1라운드 함수는 서로 다른 군에 속하는 연산을 사용하여 라운드의 각 출력 비트가 평문의 모든 비트와 라운드에 사용된 모든 비트에 의존하게 한다. 즉 F1은 mod $2^{16}+1$ 곱셈과 mod 2^{16} 덧셈 연산을 사용하여 두 개의 16비트 입력 서브 블록이 두 개의 16비트 키 서브블록에 의해 제어되며 두 개의 16비트 출력 서브블록으로 변환되어 완전한 diffusion을 달성한다. F2함수는 암호학적으로 강한 부울 함수를 사용하여 안전성을 향상시킨다. 암호학적으로 강한 부울함수의 세 가지 성질중에서 첫 번째 성질은 Webster와 Tavares[8]가 소개한 SAC(Strict Avalanche Criteria)를 만족해야 한다는 것이다. 이러한 성질은 입력 비트가 1비트가 차이가 날 때 출력 비트는 최소한 2비트이상 차이가 나는 성질로 암호학적인 함수가 SAC을 만족하면 각 출력비트는 입력비트의 한 비트가 변하면 1/2의 확률로 변한다. 두 번째 성질은 암호 알고리즘이 비선형이어야 한다는 것인데 이미 기본적으로 널리 받아들이는 성질이다. Meier와 Staffelbach[9]는 부울함수에 대한 비선형성이 암호학적인 설계에 대해 적합하다는 것을 증명하였다. 세 번째 성질은 Adam과 Tavares[10]가 제안한 부울함수가 기본적으로 0/1 balance가 될 필요가 있다는 것이다.

위에서의 세 가지 성질들을 만족하는 암호학적으로 강한 부울 함수로 구성된 라운드 함수와 가변적인 순환쉬프트를 사용함으로써 DC와 LC에 대해 안전성을 제공한다[11]. 설계한 알고리즘에 대한 안전성을 평가하기 위해서 통계적 랜덤 검정과 Avalanche Effect 검정의 결과로써 안전성을 확인한다[12].

4.1 통계적 랜덤 테스트

블록 암호를 CBC(Cipher Block Chaining) 모드로 동작시켰을 때 출력 수열에 대해 다음의 통계적 검정으로 다음의 4가지 테스트 데이터를 사용하여 각 검정을 수행하였다. 각 data file은 1Mbit의 길이를 가진다.

<표 1> 테스트 데이터

data file	사용된 128 비트 키	사용된 128 비트 평문스트링
Data1	00000000000000000000000000000000	0his is a plaint
Data2	00000000000000000000000000000001	1his is a plaint
Data3	ffffffffffffffffffffffffffffffff	0his is a plaint
Data4	00000000000000000000000000000011	0his is a plaint

4.1.1 Frequency 테스트(빈도 검정)

대상 수열에 대해 0과 1의 수가 균일하게 분포하고 있는지를 결정하는데 사용되는 검정이다.

- 통계량의 분포 : $T \sim$ 자유도 1인 χ^2분포
  유의수준 α의 기각역 : $T > \chi^2(1,\alpha)$
- test 결과 (α = 5% , $\chi^2(1,\alpha)$ = 3.841)

<표 2> Frequency 테스트 결과

data file	χ^2 value	검정 결과
Data1	0.1603 (n_0 = 524083, n_1 = 524493)	Pass
Data2	1.1878 (n_0 = 523730, n_1 = 524846)	Pass
Data3	0.4835 (n_0 = 523932, n_1 = 524644)	Pass
Data4	2.3627 (n_0 = 523501, n_1 = 525075)	Pass

4.1.2 Serial 테스트(계열 검정)

대상 수열에서 00, 01, 10, 11의 비트쌍이 고려되어지는 검정으로 검증하려는 대상 이진 수열에서 한 비트가 그 다음 비트로 가는 전이확률을 검증하는 것이다.

- 통계량의 분포 : T ~ 자유도 2인 χ^2분포
  유의수준 α의 기각역 : $T > \chi^2(2,\alpha)$t
- test 결과 (α = 5% , $\chi^2(2,\alpha)$ = 5.991)

<표 3> Serial 테스트 결과

data file	χ^2 value	검정 결과
Data1	2.1516(n_{00}=262300 , n_{01}=261783 , n_{10}=261782 , n_{11}=262710)	pass
Data2	1.4730(n_{00}=261724 , n_{01}=262006 , n_{10}=262006 , n_{11}=262839)	pass
Data3	1.3038(n_{00}=261556 , n_{01}=262376 , n_{10}=262376 , n_{11}=262267)	pass
Data4	3.2398(n_{00}=261117 , n_{01}=262383 , n_{10}=262384 , n_{11}=262691)	pass

4.1.3 Poker 테스트(포커 검정)

대상 수열에서 임의의 m 비트의 패턴을 고려하는 검정으로서 2^m의 서로 다른 패턴이 존재한다.

- 통계량의 분포 : T ~ 자유도 2^m-1인 χ^2분포
  유의수준 α의 기각역 : $T > \chi^2(2^m-1,\alpha)$
- test 결과 (m=5, α = 5% , $\chi^2(31,\alpha)$ = 44.9853)

<표 4> Poker 테스트 결과

data file	χ^2 value	검정 결과
Data1	25.9161	Pass
Data2	39.1901	Pass
Data3	34.1428	Pass
Data4	23.2150	Pass

4.1.4 Runs 테스트(런 검정)

수열에서 '0'이나 '1'이 연속하여 나타나는 것을 run이라 한다. 이 검정은 수열에서 다양한 길이의 run들의 수가 난수에 대해 예측되어지는 것과 같은지를 결정한다.

- 통계량의 분포: T ~ 자유도 $2(L-1)$인 χ^2분포
  유의수준 α의 기각역 : $T > \chi^2(2(L-1),\alpha)$
- test 결과 (L=15, α = 5% , $\chi^2(28,\alpha)$ = 41.34)

<표 5> Run 테스트 결과

data file	χ^2 value	검정 결과
Data1	23.6430	Pass
Data2	29.4638	Pass
Data3	31.4948	Pass
Data4	25.4448	Pass

4.1.5 Autocorrelation 테스트(자기상관 검정)

이진 수열 S_n이 주어졌을 때 (S_n)에서 d 비트만큼 전이시켜 생성한 수열 (S_{n+d})과의 상관 관계를 조사하는 검정이다.

- 통계량의 분포 : T ~ 자유도 2인 χ^2분포
  유의수준 α의 기각역 : $T > \chi^2(2,\alpha)$
- test 결과 (α = 5% , $\chi^2(2,\alpha)$ = 5.991)

<표 6> Autocorrelation 테스트 결과

data file	d=3, χ^2 value	d=7, χ^2 value	검정 결과
Data1	2.1336	2.7097	Pass
Data2	5.6538	2.2305	Pass
Data3	1.6593	2.0641	Pass
Data4	3.3680	4.5009	Pass

4.2 Avalanche Effect 테스트

키와 평문이 1 비트씩 바뀌었을 때 4 라운드까지의 avalanche effect를 보이며 테스트는 128비트의 각 비트 위치를 전부 한번씩 변경시켜 수행하였다.

<표 7> Avalanche effect 테스트 결과

	1라운드 후	2라운드 후	3라운드 후	4라운드 후
평문 고정, 키 1비트 변경	16.6%	55.2%	53.6%	47.8%
키 고정, 평문 1 비트 변경	27.4%	48.8%	50.8%	48.9%

실험 결과 2 라운드 후에는 완전한 avalanche effect

를 보이는 것을 알 수 있다.

5. 알고리즘 동작 속도에 대한 성능 비교 테스트

테스트 환경은 CPU는 Pentium Pro 180MHz, 메모리는 64MBytes, 컴파일러는 DJGPP 2.01를 사용해서 평가하였을 경우, 128 비트 입력 키에 대한 각 라운드의 수행 속도는 <표8>과 같다.

<표 8> 라운드에 따른 성능 테스트

라운드 수	성능(Mbps)
1	27.30
4	10.92
8	6.06
12	3.90
16	2.87

또한, 다른 블록 알고리즘과 성능을 비교해 보았을 때 그 결과는 <표 9>와 같이 수행 속도면에서는 IDEA나 RC5와는 비슷하다는 것을 알 수 있다.
비록 기존의 블록 암호 알고리즘과 같은 수행속도를 나타내었지만 알고리즘을 최적화시키고 어셈블러 수준으로 작성할 경우 수행 속도를 증가시킬 수 있다. 특히, 이를 하드웨어로 구현하여 custom chip화 할 경우 상당히 고속의 암호화가 적용 가능하리라 기대된다.

<표 9>다른 블록 암호알고리즘과의 성능 비교

알고리즘	블록 크기	키 크기 (bits)	라운드 수	처리속도 (Mbps)
DES	64	64	16	15.9
IDEA	64	128	8	15.9
RC5	64(가변)	128(가변)	12(가변)	22.75
제안	128	128(가변)	8(가변)	6.06

6. 결론

본 논문에서는 효율적이고 안전한 새로운 블록 암호 알고리즘을 제안하였다. 제안된 블록 암호는 비교적 간단한 연산만을 사용하므로 소프트웨어와 하드웨어 구현이 효율적이다. 또한 128비트에서 256비트까지의 가변적인 키 길이와 가변적인 라운드 수를 가지므로 안전성의 요구에 따라 키 길이와 라운드 수를 조절할 수 있다. 각 라운드는 2단계로 나누어지고, 서로 다른 2개의 F함수를 사용하는 이중 인벌루션 구조를 가지며, 다른 2개의 키 스케줄링을 사용함으로써 기존의 알려진 공격에 대해 안전성을 제공한다. 제안된 알고리즘의 가장 큰 특징은 가변성을 제공함으로써 다양한 정보원으로부터 추출된 멀티미디어 정보의 암호화에 정보원 서비스별로 알고리즘을 적용함으로써 서비스 수준에 맞는 알고리즘의 안전성을 제공할 수 있으리라 기대된다.

참고문헌

[1] J.L. Massey, "SAFER K-64: A byte-oriented block-ciphering algorithm", R. Anderson, editor, Fast Software Encryption, Cambridge Security Workshop(LNCS 809), 1-17, Springer-Verlag, 1994

[2] J.L. Massey, "SAFER K-64: One year later", B. Preneel, editor, Fast Software Encryption, Second International Workshop(LNCS 1008), 212-241, Springer-Verlag, 1995

[3] M. Matsui, "New Block Encryption Algorithm MISTY", R. Anderson, editor, Fast Software Encryption, Cambridge Security Workshop, Springer-Verlag, 1997, pp.53-67

[4] B. Schneier, "Applied Cryptography", John Wiley & Sons, 1996

[5] E. Biham, A. Shamir, "Differential Cryptanalysis of DES-like cryptosystems", Journal of Cryptology, vol. 4, no.1 1991, pp.3-72

[6] M. Matsui, "Linear Cryptanalysis method for DES cipher", EUROCRYPT'93, LNCS. vol.765, Springer-Verlag, 1993, pp.386-397

[7] R. Rivest, "The RC5 Encryption Algorithm", Fast Software Encryption'94, LNCS, vol. 1008, Springer-Verlag, 1994, pp.86-96

[8] A. F. Webster and S. E. Tavares, "On the Design of S-boxes", Proc. of CRYPTO'85, Springer-Verlag, 1985

[9] W. Meier and O. Staffelbach, "Nonlinearity Criteria for Cryptographic Functions", Proc. of EUROCRYPTO'89, 1989

[10] C. Adam and S. E. Tavares, "The Structured Design of Cryptographically Good S-boxes", J. of Cryptology, 1990

[11] X. Lai, "On the design and security of block ciphers", ETH Series in Information Processing, J.L. Massey (editor), vol. 1, Hartung-Gorre Konstanz, Technische Hochschule (Zurich), 1992

[12] Knuth, Donald Ervin, "The Art of Computer Programming", 2nd ed., Addison-Wesley, 1981