通信情報保護學會論文誌
第 6 卷 第 4 號, 1996. 12

안전성이 증가된 DES형 알고리즘의 제안

정 석 원* , 김 희 진* , 임 종 인* , 서 창 호**

A proposal on DES-like algorithm which increases the security

Seok-Won Jung*, Hee-Jean Kim*, Jong-In Lim*, Chang-Ho Seo**

요 약

본 논문에서는 DES의 내부 함수인 확장 함수 E(E expansion), 순환 함수 P(P permutation), S 박스와 키 생성 알고리즘(key schedule algorithm)을 그대로 사용하면서 modular 덧셈을 추가하여 최소로 변형된 DES형 알고리즘을 제안한다. 실험 결과 실험 속도 면에서 DES와 별 차이가 없지만 DC 및 LC에 대한 안전도는 증가된 것으로 추정된다.

Abstract

In this paper, a simple method to increase securities of DES against DC and LC will be proposed. Our implementation results that the proposed algorithm is as fast as DES.

1. 서 론

정보화 사회로 진입한 오늘날 정보 보호 문제는 전통적 요구인 신뢰성(confidentiality)은 물론 무결성(integrity), 인증(authentification) 등 다양한 서비스를 요구하고 있다. 이러한 요구 조건을 만족시키기 위해 흔히 암호 기법이 사용되고 있고, 현재 가장 널리 「특히 금융 분야에서」 쓰이는 암호 기법은 1977년 발표된 DES이다. Feistel Network 기법에 의하여 설계된 DES는 효율성과 NIST에 의한 FIPS (연방 정보 처리 표준)로의 지정 때문에 널리 사용되어 왔지만 56 비트라는 키의 크기와 DES 내부 구조 설계 원칙의 비공개 등 때문에 끊임없이 안전성을 의심받아 왔고 따라서 라운드 함수, 키 생성 알고리즘, 라운드 수 등에 변화를 준 수많은 DES형 블록 암호 알고리즘들이 제안되어 왔다.

* 고려대학교 수학과
* 한국전자통신연구소

그러나 주기와 선형복잡도 등으로서 안전도를 측정할 수 있는 스트림 암호의 경우와는 달리 블록 암호의 경우에는 효율적이고 정형화된 안전도 평가 방법이 존재하지 않았다. 이러한 상황은 1990년대 초 Biham과 Shamir에 의한 입출력 변화 공격법(DC, differential cryptanalysis)[2]과 Matsui에 의한 선형 공격법(LC, linear cryptanalysis)[11]에 의해 타개되었다. 이들 공격법을 적용한 결과 FEAL, GDES, s^2DES 등이 취약점을 노출하였으며 많은 블록 암호 알고리즘들도 보완의 필요성을 느끼게 되었다.

현재의 추세는 IDEA의 경우에서와 같이 대수적 성질이 다른 연산들을 사용하여 비선형성, confusion 효과를 달성하여 안전도를 강화하는 방법, Luby-Rackoff 암호에서와 같이 안전한 원시 암호 함수(cryptographic primitive)를 사용하여 안전한 암호 알고리즘을 설계하는 방법[10], unbalanced Feistel Network 개념에서와 같이 다양하게 일반화된 Feistel Network 개념을 도입하여 복잡도를 측정함으로서 provable secure한 암호 알고리즘을 설계하는 방법[16] 등이 있다. 그러나 위에서 제안된 암호 알고리즘들을 하드웨어로 구현하는 것은 몇 가지 어려움이 예상된다. 또한 현재 DES의 보급 상황이나 DES 칩(chip)의 개발 정도를 볼 때 DES 알고리즘에 약간의 변화를 가하여 적은 비용으로 DES 알고리즘을 보완함으로서 안전도를 증가시키는 것은 의미가 있다고 생각한다.

본 논문은 이와 같은 목적을 가지고 S 박스 투입 과정에 약간의 변화를 주었을 때 발생되는 안전도의 증가에 대해 살펴보았다. 2가지 방법을 생각해 보았으며 이것들은 2절에 수록되어 있다. 3절에는 제안된 2가지 방법 중 56 비트 키를 그대로 사용하였을 경우에 대한 실험 결과를 수록하고 있다. 암호화,복호화 시간은 DES의 경우와 별 차이가 없었으며 DC 및 LC에 대한 안전도는 증가된 것으로 추정된다. 향후 본 논문에서 제안된 DES형 알고리즘에 대한 보다 철저한 분석을 계속할 예정이다.

2. 새로운 블록 알고리즘의 제안

2.1 암호화 알고리즘

전체 구조는 기존의 DES와 같다. DES에서 사용되는 확장 함수 E와 순환 함수 P, S 박스 그리고 키 생성 알고리즘을 그대로 사용하였다. 다만 다음과 같이 각 단(round)에 modular 덧셈을 한가지 더 추가한다. 한 단에 대한 알고리즘은 그림 1과 같다. 64 비트 입력을 둘로 나누어 왼쪽 반을 L, 오른쪽 반을 R로 표시하자. 그리고 키 생성 알고리즘에 따라 48 비트 키 K가 각 단마다 생성된다. i 단은 다음의 식으로 요약할 수 있다.

$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$$

함수 F는 그림 2에 나타나 있다. 이 i 단 키 K_i를 32 비트 부분 키 K_i^1과 16 비트 부분 키 K_i^2로 나눈다. 그리고 32 비트 입력 R_{i-1}과 32 비트 부분 키 K_i^1을 XOR하고 확장 함수 E로 48 비트로 확장한다. 이 48 비트는 6 비트 8개로 나누어 S 박스에 들어갈 준비를 한다. 이 데이터를 8개로 연접한 $E_1 \| E_2 \| E_3 \| E_4 \| E_1' \| E_2' \| E_3' \| E_4'$으로 본다. 이제 16 비트 부분 키 K_i^2를 두 개의 8 비트 부분 키 K_i^{21}과 K_i^{22}로 나눈다. 각 8 비트 키에 대해서 첫 비트와 마지막 비트로 만든 2 비트가 십진수 표현으로 a와 b라 하자. 각 수 a, b에 대해서 데이터 E_a와 E_b'를 고른다. 이제 이 두 데이터와 각 8 비트 부분 키의 가운데 6 비트는 덧셈 군 Z_{2^6}의 원소로 보아 덧셈을 한다. 즉, $E_a + (K_i^{21} \gg 1) \bmod 2^6$, $E_b' + (K_i^{22} \gg 1) \bmod 2^6$을 한다. 여기서 '≫' 은 한 비트 오른쪽 쉬프트를 의미한다. 이 결과로 나온 6 비트 8개 데이터는 S 박스로 들어간다. 이후의 알고리즘은 DES와 같다.

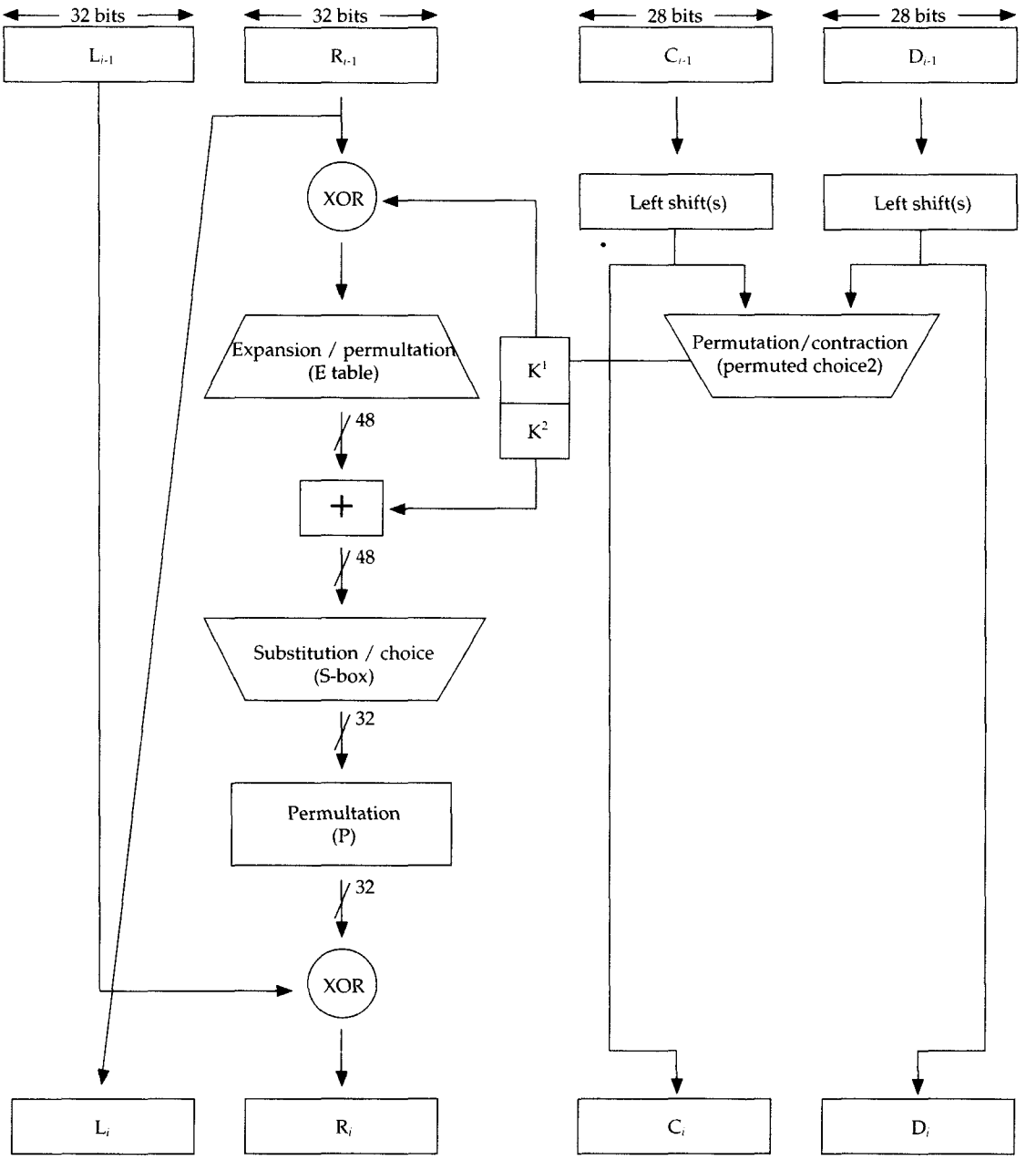
32 bits
32 bits
28 bits
28 bits
L_{i-1}
R_{i-1}
C_{i-1}
D_{i-1}
XOR
Left shift(s)
Left shift(s)
Expansion / permultation
(E table)
K^1
K^2
Permutation/contraction
(permuted choice2)
48
+
48
Substitution / choice
(S-box)
32
Permultation
(P)
32
XOR
L_i
R_i
C_i
D_i

그림 1. 알고리즘의 한 단

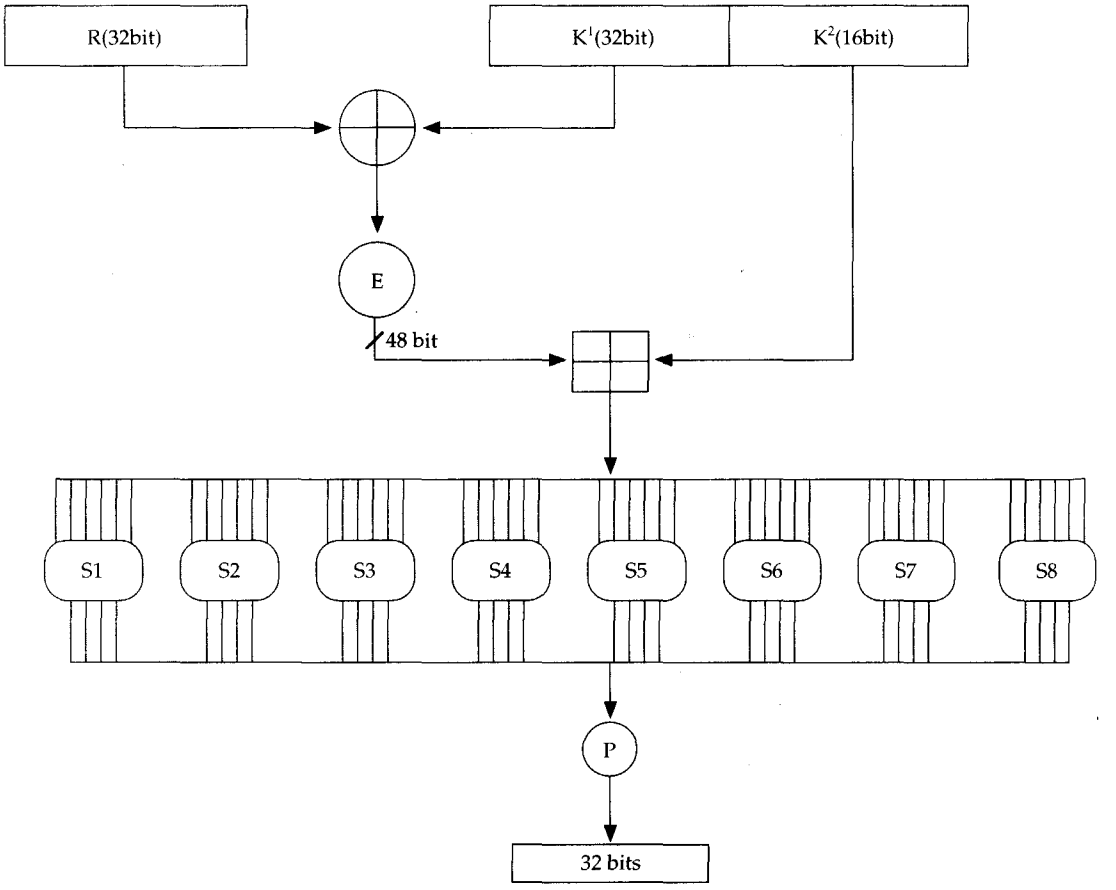

그림 2. 함수 F(R,K)의 계산

2.2 복호화 알고리즘

복호화 과정은 암호화 과정과 근본적으로는 같다. 다만 각 단에 사용되는 비밀키가 암호화에 사용된 키의 역순으로 사용된다는 점만 다르다. 암호화 과정의 마지막 단 이후 출력의 반들은 위치를 바꾸므로 최종 역초기순환(inverse initial permutation) IP^{-1} 단계에서의 입력은 $R_{16} \| L_{16}$이다. 이 단의 출력이 암호문(ciphertext)이다. 이제 이 암호문을 택해서 복호화 알고리즘에 넣자. 우선 암호문을 초기순환(initial permutation) IP 시킨 결과를 $L_0^d \| R_0^d$로 표시하자. 여기서 d는 복호화를 의미한다. 그런데 는 IP의 역이므로 다음과 같은 사실을 얻는다.

$$L_0^d \| R_0^d = IP(ciphertext),$$
$$ciphertext = IP^{-1}(R_{16} \| L_{16}),$$
$$L_0^d \| R_0^d = IP(P^{-1}(R_{16} \| L_{16})) = R_{16} \| L_{16}.$$

따라서 복호화 첫 단의 입력은 암호화 16단 출력의 32 비트들의 위치 바꿈과 같다. 우선 16단의 암호화 과정을 보자. 이는 다음과 같다.

$$L_{16} = R_{15},$$
$$L_{16} = R_{15} \oplus PS(E(R_{15} \oplus K_{16}^{-1}) + K_{16}^{-2}).$$

여기서 P, S 그리고 E는 각각 순환 함수

(permutation), S 박스 함수 그리고 확장함수(expansion)를 뜻하고 '+'는 암호화 과정에서 설명한 Z_{2^6} 위에서의 modular 덧셈이다. 이제 복호화를 살피자.

$$\begin{aligned} L_1^d &= R_0^d = L_{16} = R_{15}, \\ R_0^d &= L_0^d \oplus PS(E(R_0^d \oplus K_{16}^{-1}) + K_{16}^{-2}) \\ &= R_{16} \oplus PS(E(R_{15} \oplus K_{16}^{-1}) + K_{16}^{-2}) \\ &= [L_{15} \oplus PS(E(R_{15} \oplus K_{16}^{-1}) + K_{16}^{-2})] \\ &\quad \oplus PS(E(R_{15} \oplus K_{16}^{-1}) + K_{16}^{-2})) \\ &= L_{15}. \end{aligned}$$

그러므로 복호화 과정에서 첫 단의 출력은 $L_{15} \| R_{15}$이다. 따라서 이러한 과정을 계속 반복 수행하면 복호화 마지막 단의 출력은 $L_0 \| R_0$이다. 그런데

$$IP^{-1}(L_0 \| R_0) = IP^{-1}(IP(plaintext)) = plaintext$$

이므로 우리는 평문(plaintext)를 얻을 수 있다.

2.3 또 다른 제안

2.2절에서 제안한 암호화 알고리즘은 2개의 S 박스를 택해서 modular 덧셈을 했다. 이 절에서는 키의 크기를 늘려 여러 개의 S 박스에 modular 덧셈을 추가하는 방법에 대해서 서술하겠다. DES의 키는 64 비트에서 8개의 부호 비트를 뺀 56 비트 키에 대해서 한 번 또는 두 번의 쉬프트(shift)를 반복적으로 사용하여 각 단에 필요한 56 비트 키를 생성한 후 이 중에서 48 비트만을 선택하여 부분 키로 쓴다. 그러나 우리는 부호 비트까지 합친 64 비트 전체를 키로 사용한다. (따라서 입력은 16진수로 표현해서 넣는다.) 이에 따라 키 생성 알고리즘에 대한 약간의 수정이 필요하다. 64 비트 전체 키에 대하여 DES와 같이 한 번 또는 두 번의 쉬프트(shift)를 사용하여 각 단에 필요한 64 비트 키를 생성한다. 이 64 비트 키 중에서 DES 키 생성때 처럼 48 비트 부분 키「이는 입력과의 XOR에 사용된다」를 만든다. 그리고 또 한 번은 64 비트 키를 8개의 8 비트 키로 나누어 각 8 비트에서 최상위 비트의 값으로 S 박스에 들어가기 전의 입력과 8 비트 키 중 가운데 6 비트에 대한 modular 덧셈을 할 것인 지를 결정한다. 즉, 최상위 비트가 0이면 키의 가운데 6 비트에 대한 modular 덧셈을 수행하지 않고, 최상위 비트가 1이면 modular 덧셈을 한다. 이 방법에 따르면 각 단에서 평균적으로 4개의 S 박스에 modular 덧셈이 더 수행된다. 이에 따른 암·복호화 과정은 2.1절과 2.2절에서 설명한 것과 거의 같다.

3. 몇 가지 실험 및 성질

3.1 Avalanche 효과

블록 암호 알고리즘에서 입력에 대한 조금의 변화가 출력에 많은 변화를 주어야 함은 가장 기본적인 성질이다. 이러한 성질 중 입력에 대한 한 비트의 변화가 출력에 어느 정도 영향을 미치는 가를 측정하는 것이 avalanche 효과이다. DES의 S 박스는 이 성질에 상당히 강하다는 것은 잘 알려진 사실이다.

X와 X^*를 한 비트만 다른 입력이라 하자. 그러면 $X \oplus K^{-1}$와 $X^* \oplus K^{-1}$는 적어도 한 비트가 다르며 $E(X \oplus K^{-1})$와 $E(X^* \oplus K^{-1})$도 한 비트는 다르다. 이 두 값을 덧셈군 Z_{2^6} 위의 원소로 생각하면 두 값은 서로 다른 값이고 따라서 $E(X \oplus K^{-1}) + K^{-2}$와 $E(X^* \oplus K^{-1}) + K^{-2}$ 역시 서로 다른 값이다. 따라서 이 두 값은 적어도 한 비트가 다르다. 그러므로 avalanche 효과는 DES 보다는 약해지지 않는다.

다음의 실험 결과는 두 평문

00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000,

10000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

과 비밀 키

00000001 00100011 01001001 01100111 10001001 10101011 11001101 11101111

을 DES와 제안된 알고리즘에 사용하여 얻은 결과이다.

표 1. DES와 제안된 알고리즘에서의 avalanche 효과 비교

단	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
DES에서 달라진 비트 수	1	6	21	28	27	30	32	29	28	25	26	32	33	33	33	34	33
제안된 알고리즘에서 달라진 비트 수	1	6	22	35	35	34	34	35	34	36	34	36	38	33	34	34	32

3.2 암 · 복호화 속도 비교

블록 암호 기법은 이들보다 안전한 공개 키 암호 기법보다 암 · 복호화를 할 때 속도가 빠르며 구현이 쉬워서 많이 사용되고 있다. 특히 DES는 기본 연산을 비트 사이의 XOR로 택하고 있어 암 · 복호화 속도가 상당히 빠르다. DES는 32 비트 입력이 확장 함수 E로 48 비트로 확장된 뒤에 48 비트 키와 XOR 된다. 우리가 제안한 방식은 입력 32 비트와 키 32 비트의 XOR연산 뒤 부분적으로 덧셈 군 Z_{2^n} 위의 두 원소의 덧셈을 두 번 계산한다. 따라서 DES와의 속도 차이는 8 비트 두 원소의 XOR와 6 비트 두 원소의 modular 덧셈에서 생긴다. 따라서 속도의 차이가 별로 없음이 예측되며 실제로 다음의 실험 결과에서 보듯이 두 가지 연산은 큰 차이가 없다.

표 2. DES와 제안된 알고리즘의 암 · 복호화 속도 비교

자료크기 \ 알고리즘		8단 DES	8단 제안 알고리즘	16단 DES	16단 제안 알고리즘
1000X8 bytes	암호	3.57초	3.51초	4.62초	4.56초
	복호	3.57초	3.52초	4.55초	4.50초
10000X8 bytes	암호	35.81초	35.09초	46.52초	45.37초
	복호	35.70초	35.43초	46.19초	45.53초
100000X8 bytes	암호	392.00초	385.19초	489.60초	489.46초
	복호	388.43초	382.45초	489.33초	489.06초

〈참고〉 1) 여기에 사용된 DES는 알려진 최적 프로그램을 사용하지는 않았다. (최적 source 코드는 Schneier의 Applied cryptography, second edition[15]에서 얻을 수 있다.) 우리는 DES 알고리즘을 직접 code 했고(따라서 효율은 떨어짐), 이 DES 알고리즘을 변형하여 제안된 알고리즘을 code 했다. 그러므로 시간의 차이는 앞에서 설명하였듯이 8 비트 XOR 연산과 6 비트 modular 덧셈에서만 생긴다.

2) 구현은 IBM 486 PC 66Mhz 에서 Turbo C 3.0을 사용하여 실행하였다. 시간은 자료 파일로부터 자료를 읽고 암호화 또는 복호화 한 후 자료를 파일에 저장하는데 까지 걸린 시간이다.

3.3 DC와 LC에 대한 성질

부분적인 modular 덧셈에 대해서는 다음과 같은 성질이 성립함을 쉽게 알 수 있다.

1) 이 연산은 XOR연산과 배분이나 결합 법칙이 성립 안 한다. 즉,

$$(X \oplus X^*) + K \neq (X + K) \oplus (X^* + K)$$
$$(X \oplus X^*) + K \neq X \oplus (X^* + K)$$

이다.

2) 이 연산에 대해 입력의 선형 변화가 출력의 비선형 변화를 일으킨다.

3) 키의 효과를 없애는 입력에 대한 차이(input difference)를 구하기 어렵다. 즉, 제안된 방식은

부분적으로 XOR와 modular 덧셈이 사용되었으므로 입력의 차이를 $\Delta X = X \oplus X^*$ 또는 $\Delta X = X - X^*$로 정의해도 비밀 키 K 의 효과를 없앨 수 없다.

위의 성질들로부터 제안된 알고리즘은 DC와 LC에 대해 적용이 어려움을 다음과 같이 알 수 있다.

DES에서는 두 입력 X와 X^*에 대해서

$$(E(X) \oplus K) \oplus (E(X^*) \oplus K) = E(X) \oplus E(X^*) = E(X \oplus X^*)$$

이므로 두 입력의 차이를 $\Delta X = X \oplus X^*$라 하면 비밀 키 K의 효과를 없앨 수 있다. 따라서 입력의 차에 대한 S 박스의 출력 사이의 상관을 살펴 확률이 높은 characteristic을 구성한다. 이에 따라 특정한 형태, 즉 고정된 입력 차에 대응하는 입력 쌍 X, X^*와 출력의 차이 $Y' = Y \oplus Y^*$에 대하여

$$S(X \oplus K) \oplus S(X^* \oplus K) = Y' \quad \text{-- 식 (1)}$$

를 만족하는 비밀 키 K를 구하는 방법이 DC이다. 그런데 제안된 알고리즘에서는 부분적으로 modular 덧셈이 한번 더 적용되므로 modular 덧셈이 안 쓰인 곳에서는 $\Delta X = X \oplus X^*$를 입력 차이로 하면 DES와 같은 characteristic을 사용해서 키의 몇 비트를 구할 수 있다. 그러나 modular 연산이 사용된 곳에서는 입력의 차이를 $\Delta X = X \oplus X^*$로 택하면

$$[E(X \oplus K^{-1}) + K^{-2}] \oplus [(E(X^* \oplus K^{-1}) + K^{-2}] \neq E(X \oplus X^*)$$

이므로 키 K를 제거할 수 없다. 그런데 입력의 차이를 $\Delta X = X - X^*$로 정의하면

$$[E(X \oplus K^{-1}) + K^{-2}] - [(E(X^* \oplus K^{-1}) + K^{-2}] = E(X \oplus K^{-1}) - E(X^* \oplus K^{-1})$$

이므로 키 K^{-2}를 지울 수 있으므로 부분적으로는 characteristic을 구성할 수 있다. 그러나 제안된 알고리즘에서는 비밀키에 따라 어느

부분이 modular 덧셈을 할 지가 결정되므로 전체 입력에 대한 차이를 정의하기 어렵다. 따라서 최적 characteristic의 구성이 어려워지고 DC 공격법이 어려워진다.

다음은 LC에 대해서 살펴보자. DES에 대하여는 S-box에 있어서 입력에 대한 몇 비트의 XOR와 출력에 대한 몇 비트의 XOR는 유의 확률 p($\neq$ 1/2)로 같아진다. 즉, 유의 확률 p로 다음 식이 성립한다.

$$(X \oplus K)[i_1, i_2, \cdots, i_a] = Y[j_1, j_2, \cdots, j_b].$$

여기에서 $[i_1, i_2, \cdots, i_a]$는 $i_1, i_2, \cdots, i_a$번째 비트들의 XOR를 뜻한다. 그런데 XOR 연산은 비트 사이에 교환법칙이 성립하므로 다음과 같은 선형근사식을 얻을 수 있다.

$$X[i_1, i_2, \cdots, i_a] \oplus Y[j_1, j_2, \cdots, j_b] = K[l_1, l_2, \cdots, l_c].$$

이 식에 대해 maximum likelihood 법을 이용하여 비밀 키 $K[l_1, l_2, \cdots, l_c]$의 한 비트를 추정하는 것이 LC의 개요이다. 그런데 우리가 제안한 알고리즘에서는 modular 덧셈이 XOR와 결합이 되지 않으므로 식

$$((X \oplus K^1) + K^2)[i_1, i_2, \cdots, i_a] = Y[j_1, j_2, \cdots, j_b]$$

에 대하여 선형 근사식 구성이 어렵다. 따라서 제안된 알고리즘은 DES에 비해서는 LC에 대한 공격 법에 강할 것으로 추정된다.

3.4 입출력 변화 공격법에 대한 실험

우선 4 단(round)에 대해서 알아보자. 4 단에 대한 characteristic은 DES 4 단 characteristic을 사용하자. DES 4 단에 대해서는 평문의 차이 ΔX = 20 00 00 00 00 00 00 00(16 진수 표현)를 사용하여 확률 1인 characteristic를 구성한다. 이는 우리가 제안한 알고리즘에도 확률 1로 성립한다.

이제 평문들을 어떻게 만들었는 지에 대해서 알아보자. 평문의 64 비트이므로 우리는 이를 $GF(2^{64})$ 위의 원소로 볼 수 있다. 따라서 원시다항식을 $p(x) = x^{64} + x^4 + x^3 + x^2 + 1$ 라 놓으면 갈로아 체 $GF(2^{64}) = GF(2)[x] / (p(x))$으로 볼 수 있다. 이때 x가 원시원이므로 x의 멱승으로 모든 원소를 표현할 수 있는데 x에 대한 $GF(2^{64})$의 다항식 기저로의 표현은 63 비트가 0이므로 우리는 임의의 원소를 택하여(원시원이 될 확률이 거의 1이다) 이 원소의 멱승으로 평문들을 구하였다.

사용된 키는 16 진수 표현으로 0123456789 abcdef 이고 이들에 대한 4 단 부분 키는

$$E(K_6^1) = 2e\ 25\ 11\ 13\ 3a\ 25\ 14\ 0a\ ,$$
$$K_4^2 = 82\ 57$$

이다. 이 때 $K = E(K_4^1)$로 놓고 식 (1)을 만족하는 키를 구해보면 S_2, S_4, S_5, S_7 그리고 S_8에 대한 옳은 키를 5개의 암호문 쌍을 사용하여 구할 수 있었다. 또한 caracteristic이 1이므로 S_3과 S_6에 관해서는 모든 키가 식 (1)을 만족하지 못한다. 즉, 이 S 박스들에 modular 덧셈이 사용되었음을 알 수 있다. 따라서 우리가 알아낸 비트의 수는 30 비트이다. 이는 4 단 DES가 42 비트의 키를 찾을 수 있는 것에 비해 상당히 적은 비트를 구한 것이다.

다음은 6 단에 대해서 알아보자. 6 단 DES의 최적 characteristic은 평문의 차이 ΔX = 40 08 00 00 04 00 00 00(16 진수 표현)를 사용한 것이다. 이 때 확률은 1/16로 120개의 암호문 쌍을 이용하여 S_2, S_5, S_6, S_7, S_8에 대한 비밀 키 30 비트를 구할 수 있다. 그런데 제안된 알고리즘에서는 DES에서 사용된 charcteristic이 modular 덧셈이 두 개의 S 박스에 사용되므로 일반적으로 최적은 아니다. 비밀 키를 16 진수 표현으로 0123456789abcdef를 사용하였을 때 6 단에 사용된 부분 키는

$E(K_6^1)$ = 04 06 14 05 23 3c 07 38 ,
$E(K_6^2)$ = 83 65

이다. 따라서 modular 덧셈이 사용된 곳은 S_4와 S_6이다. 우리는 290개의 암호문 쌍을 사용하여S_2, S_5, S_7, S_8에 사용된 키를 찾았다. (이는 이 characteristic의 확률이 1/16 보다 적음을 뜻한다.) 그런데 S_6 박스에 대해서도 290개의 암호문 쌍을 사용하면 키의 후보를 하나 고를 수 있다. 그런데 식 (1)을 사용하여 얻는 이 키는 옳바른 키가 되지 않는다. 따라서 도청자는 적어도 한 번은 더 290개의 암호문 쌍을 이용하여 옳바른 키인지를 확인해야 한다. 우리의 실험 결과 S_6 박스에 대한 키로 290개의 암호문 쌍을 이용했을 때 01(16진수 표현)을 찾았고 다시 290개의 암호문 쌍을 사용하면 3d를 얻는다. 따라서 이 박스에 modular 덧셈이 사용되었음을 알 수 있다.

4. 결론과 연구 방향

본 논문은 DES의 내부 함수를 그대로 사용하면서 여기에 modular 덧셈 연산을 더 써서 DES와 유사한 암호 알고리즘을 제안하였다. 이는 DES의 내부 구조를 그대로 유지하므로 DES의 기본적인 성질들을 충분히 만족시킨다. 또한 구현 속도 면에서 보면 Z_{2^6} 위에서의 덧셈을 수행하므로 DES의 8 비트 XOR 연산과 비교하면 속도가 거의 같다. 그리고 6 단 알고리즘에 대한 DC 공격법에서 알아보았듯이 DES에서 사용한 최적의 characteristic이 우리가 제안한 알고리즘에서는 최적이 되지 않으며 확률도 알 수 없다. 따라서 DC를 적용시킬 때 DES에서 사용한 characteristic을 써도 확률은 최소로 추정하여 그에 대한 암호문 쌍의 수를 결정해야 한다. 또한 modular 덧셈이 사용되었는 지를 알기 위해서는 두 번 이상의 테스트를 실시해야 한다. 이렇게 하여도 DC를 DES에 적용시켰을 때 보다는 적은 비트에 대한 비밀 키의 정보를 얻는다.

또한 앞에서의 설명에서와 같이 LC에 대해서도 강해지리라 예상한다.

앞으로의 연구 방향은 다음과 같다.

1. 제안 알고리즘에 대한 더 높은 단에서의 DC 공격 적용, 이에 따른 16 단 DES와 같은 난도를 같는 단 수의 결정
2. LC에 대한 부분적인 적용.(modular 덧셈을 부분적으로 사용하므로 어느 정도 가능하리라 예상한다.)
3. 2,3절에서 제안된 알고리즘에 대한 속도 및 안전도 분석, 그리고 modular 곱셈을 이용한 알고리즘에 대한 분석

참 고 문 헌

[1] C.M. Adams, Designing DES-like ciphers with guaranteed resistance to differential and linear attack, to appear.

[2] E. Biham and A. Shamir, Differential cryptanalysis of the data encryption standard, New York, Springer-Verlag, 1993.

[3] E. Biham, On Matsui's linear crytanalysis, presented at Dagstuhl Seminar, 1993.

[4] E. Biham, New types of cryptanalytic attacks using related keys, J. of Cryptology, v. 7, n. 4, 1994, pp. 229-246.

[5] D. Coppersmith, The data encryption standard(DES) and its strength against attacks, IBM research report RC 18613(81421), T.J.Watson research center, December, 1992.

[6] W. Diffie and M.E. Hellman, New derections in cryptography, IEEE Trans. on Information Theory, v. IT-22, n.6, 1976, pp.644-654.

[7] C. Harpes, G.G. Kramer, J.L. Massey, A generalization of linear cryptanalysis and the applicability of Matsui's piling-up lemma, Eurocryp '95,Springer,1995, pp. 24-38.

[8] H.M. Heys and S.E Tavares, Substitution-permutation networks resistant to differential and linear cryptanalysis, J. Cryptology, 1996, pp.1-19.

[9] B.S. Kaliski and M.J.B. Robshaw, Linear cryptanalysis using multiple approximations, Advanced in Cryptology - Eurocrypt'94, Springer, 1994, pp. 26-39.

[10] S. Lucks, Faster Luby-Rackoff ciphers, Fast Software Encryption, LNCS 1039, Springer-Verlag, 1996, pp. 189-203.

[11] M. Matsui, Linear cryptanalysis method for DES cipher, Proceedings Eurocrypt'93, 1993, published by Springer-Verlag.

[12] S. Murphy, The cryptanalysis of FEAL-4 with 20 chosen plaintexts, J. of Cryptology, No.3 , 1990.

[13] K. Nyberg and L.B. Knudsen, Probable security against a differential attack, J. Crptology, 1995, pp. 27-37.

[14] B. Preneel, Analysis and design of cryptographic hash functions, Ph.D. Dissertation, Katholieke Universiteit Leuven, 1993.

[15] B. Schneier, Applied cryptography, second edition, John Wiley & Sons, 1996.

[16] B. Schneier and J. Kelsey, Unbalanced

Feistel networks and block cipher design, Fast Software Encryption, LNCS 1039, Springer-Verlag, 1996, 121-144.

[17] C.E. Shannon, Communication theory of secrecy systems, Bell System Technical Journal, v. 28, n. 4, 1949, pp. 656-715.

[18] W. Stallings, Network and internetwork security, Prentice Hall, 1995.

□ 著者紹介

정 석 원(鄭 錫 元)

1991년 2월 고려대학교 수학과 학사
1993년 2월 고려대학교 수학과 대학원 석사
1995년 2월 고려대학교 수학과 대학원 박사수료

※ 주요관심분야 : 응용 대수학 및 정수론, 암호론

김 희 진

1992년 2월 고려대학교 수학과 학사
1994년 2월 고려대학교 수학과 대학원 석사
1996년 2월 고려대학교 수학과 대학원 박사수료

※ 주요관심분야 : 응용대수학 및 정수론, 암호론

서 창 호

1990년 2월 고려대학교 수학과 학사
1992년 8월 고려대학교 수학과 대학원 석사
1996년 8월 고려대학교 수학과 대학원 박사
1996년 현재 함국전자통신연구소 선임연구원

※ 주요관심분야 : 응용대수학 및 정수론, 암호론

임 종 인

1980년 2월 고려대학교 수학과 학사
1982년 2월 고려대학교 대학원 수학과 석사
1986년 2월 고려대학교 대학원 수학과 이학박사
1986년 8월 - 현재 고려대학교 수학과 교수

※ 주관심 분야 : 응용 대수학 및 정수론, 암호론