韓國通信情報保護學會誌
第1卷, 第1號, 1991. 4

Lucifer와 그 심볼간 상호 의존도

문상재* · 이훈제**

1. 머 리 말

Lucifer는 정보 데이타를 암호화 시키는 관용 암호화 알고리즘이다. 이러한 관용 암호화 알고리즘의 대표적인 것으로 1970년대초 개발된 Lucifer[1] 외에, 1974년에 발표된 DES(data encryption standard)[2], 그리고 1986년의 일본 NTT의 FEAL－8[3]을 들 수 있다. 근래에는 회사에서도 독자적인 관용 암호화 알고리즘을 개발 발표하고 있는 형편이다[4].

관용 암호화 알고리즘은 exhaustive attack에 의한 분석, 또한 확률적 및 대수적 암호분석으로부터 보호되야 한다. 전자는 주로 키이의 크기에 관련되며, 후자에 대한 안전도 측정은 암호화된 출력의 불규칙성 및 입출력간의 심볼간 상호의존성에 의해 주로 측정된다. 입력되는 심볼의 변화에 대한 출력문의 모든 심볼의 변화를 표시하는 심볼간 상호의존은 암호화 알고리즘의 구조에 의해 결정되며, 암호화 알고리즘의 설계에 중요한 역할을 한다. 암호화 과정에서 출력비트는 초기 입력 정보비트와 열쇠비트들의 함수로 볼 수 있다. 출력비트와 어떤 입력비트의 두 비트 사이에 이러한 함수적 관계를 지니게 되면 상호 의존한다고 하고, 이러한 성질을 백분율로 나타낸 값을 의존도라 한다.

본 연구에서는 Lucifer의 암호화 과정을 자세히 설명한다. 이를 바탕으로 라운드 별로 평문과 암호문의 심볼간 의존도 및 키이와 암호문의 심볼간 의존도를 계산하고, 라운드가 거듭함에 따른 의존율의 추세를 알아본다.

* 정회원, 경북대학교 전자공학과 부교수
** 정회원, 국방과학연구소 연구원

2. Lucifer

가. 암호화 알고리즘

Lucifer는 1970년대초 미 IBM사에 의해 개발된 블록 관용 암호화 알고리즘이며, 이후 동 회사에서 이를 바탕으로 DES가 개발되어 DES의 전신이라고도 일컫는다. Lucifer에 입력되는 열쇠, 정보어 및 출력되는 암호어의 블록 크기는 똑같이 128비트이다. 이 암호화 과정은 같은 형태의 과정을 16번 반복 수행하며 한 라운드에 해당되는 암호화 과정은 그림 1과 같다.

그림 1에서 입력 정보문의 128 비트는 64 비트씩 상반부와 하반부로 나누어진다. 한 라운드가 수행되는 과정에서 상반부 및 보조 열쇠의 영향을 받아서 새로운 64 비트의 하반부가 구성되고, 이 변형된 하반부는 최종 라운드를 제외하고는 다음 라운드로 넘어가기 전에 상반부와 서로 교체되어진

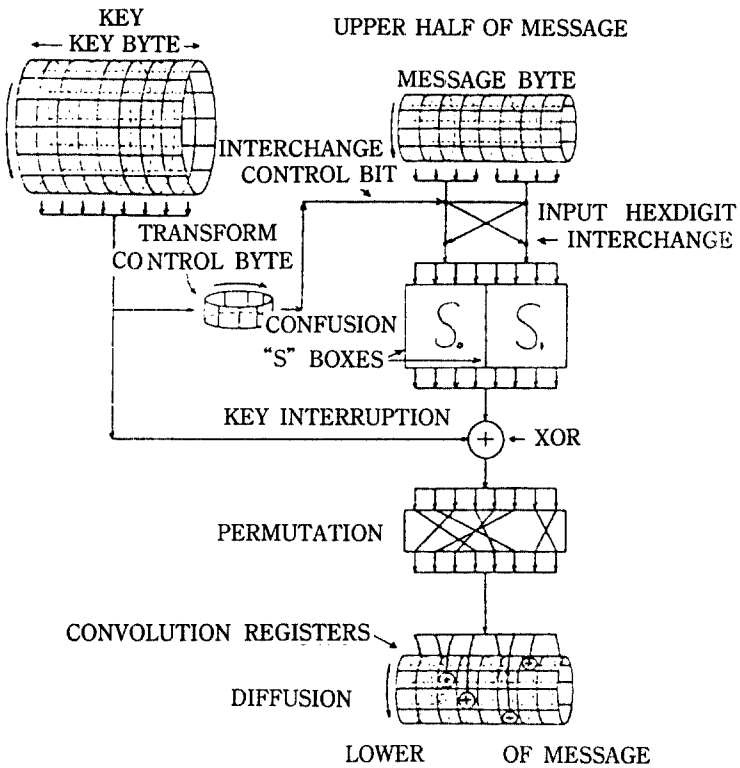

Fig. 1. Block Diagram of C-I-D(confusion-interruption-diffusion)

다.

한 라운드의 암호과정을 구체적으로 설명하면 다음과 같다. 16 바이트의 열쇠, 8 바이트의 상반부 정보 비트와 8 바이트의 하반부 정보 비트는 그림 1의 각 원통으로 표시하고, 영(zero) 바이트를 원통의 원점으로 해서 공히 같은 방향으로 회전되면서 새로운 바이트가 원점에 대치되어 암호화 과정이 수행된다. 열쇠 원통은 한 바이트씩 단계적으로 회전하지만, 매 라운드의 마지막 바이트가 수행되고 나면 그 바이트는 회전하지 않고 정지하여 다음 라운드의 첫 바이트가 된다. 표 1은 각 라운드에 사용되는 64 비트의 보조 열쇠 생성을 위한 열쇠 바이트 순서이다. 그리고 각 라운드에 사용되는 보조 열쇠의 첫 바이트는 변환 조정 비트(transform control byte)로 사용되며, 이들 8개 비트들은 상호 교환 조정 비트(interchange control bit)가 되어서 2개의 S-상자에 대한 입력 교환을 제어한다.

한 라운드 과정에서는 S_0 및 S_1의 S-상자와 하나의 P-상자가 8번 반복 수행된다. 이들은 표 2와 같다. S-상자에 대한 입력은 상호 교환 조정 비트가 "0"이면 메세지 바이트의 경우 좌우 4 비트에 의한 두 십진값이 각각 S_0와 S_1에 대한 입력이 되며, "1"이면 두 십진값이 서로 바뀌어서 들어간다.

대체를 거친 바이트는 열쇠 인터럽트 과정에서 보조 열쇠와 비트별 mod-2 연산이 행해지며, 치환과정에서 비트 자리 바꿈이 행해진다. 치환과정을 거쳐 얻어진 바이트들을 하반부의 특정한 비트와 비트별 mod-2 연산을 다시 해야 하는데 이 연산이 행해지는 형태는 그림 2와 같다.

Table 1. Key Byte Access Schedule

	0	1	2	3	4	5	6	7
1	0	1	2	3	4	5	6	7
2	7	8	9	10	11	12	13	14
3	14	15	0	1	2	3	4	5
4	5	6	7	8	9	10	11	12
5	12	13	14	15	0	1	2	3
6	3	4	5	6	7	8	9	10
7	10	11	12	13	14	15	0	1
8	1	2	3	4	5	6	7	8
9	8	9	10	11	12	13	14	15
10	15	0	1	2	3	4	5	6
11	6	7	8	9	10	11	12	13
12	13	14	15	0	1	2	3	4
13	4	5	6	7	8	9	10	11
14	11	12	13	14	15	0	1	2
15	2	3	4	5	6	7	8	9
16	9	10	11	12	13	14	15	0

Table 2. Substitution and Permutation.

S-Box Internal Permutations		Fixed Permutation
S_0	S_1	
0 …… 12	0 …… 7	0 …… 3
1 …… 15	1 …… 2	1 …… 5
2 …… 7	2 …… 14	2 …… 0
3 …… 10	3 …… 9	3 …… 4
4 …… 14	4 …… 3	4 …… 2
5 …… 13	5 …… 11	5 …… 1
6 …… 11	6 …… 0	6 …… 7
7 …… 0	7 …… 4	7 …… 6
8 …… 2	8 …… 12	
9 …… 6	9 …… 13	
10 …… 3	10 …… 1	
11 …… 1	11 …… 10	
12 …… 9	12 …… 6	
13 …… 4	13 …… 15	
14 …… 5	14 …… 8	
15 …… 8	15 …… 5	

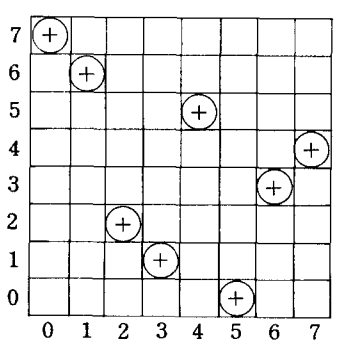

Fig. 2. Unfolded Convolutional Registers.

나. 특성 분석

암호화 과정의 매 라운드에서 출력비트는 초기 입력 정보비트와 열쇠비트들의 함수로 볼 수 있다. 출력비트와 어떤 입력비트의 두 비트 사이에 이러한 함수적 관계를 지니게 되면 상호 의존한다고 한다. 이러한 성질을 백분율로 표현한 값이 의존도이다. DES에 대한 심볼간 상호 의존도는 Meyer[5]에 의해서 연구 발표된 바 있다. 그러나 Lucifer의 암호화 과정이 다르므로 이를 직접 적용할 수 없어 앞 절

에서 해석된 내용을 바탕으로 Lucifer의 상호 의존도를 구해 본다.

1) 평문과 암호문의 심볼간 상호 의존도

그림 3은 앞에서 설명된 Lucifer의 16 라운드 암호화 과정을 상호 의존도 해석에 편리하도록 나타낸 블록선도이다. 그림 3에서 L_i와 U_i, $i=0,1,\cdots,16$는 각각 8 바이트로 구성된 하단과 상단 부블록을 나타내며, 매 라운드마다 $U_i=h(U_{i-1}, K_i)\cdot L_{i-1}$의 암호화 과정이 수행된다. 여기서 K_i, $i=1,2,\cdots,16$는 i번째 라운드에 사용하기 위해 생성된 열쇠이며, ⊕는 비트별로 행해지는 mod−2를 나타낸다.

그림 3의 매 라운드마다 사용되는 함수 h의 내부 구조를 알기 쉽게 나타낸 것이 그림 4이다. U_{i-1}의 바이트들은 열쇠의 제어하에 S_0와 S_1 상사에 의해 대체과정을 거쳐 열쇠 K_i와 mod−2로 덧셈한 후, P 상자에서 치환이 된 다음에 다시 아래 표 3에 의해 치환이 되어 U_i가 된다. i 라운드 출력 $X_i=(L_i, U_i)$가 j 라운드(i>j) 출력 $X_j=(L_j, U_j)$에 대한 의존상태는 128×128 행렬 G_{ij}로 표시할 수 있다. G_{ij} 행렬의 (l, m)째 원소를 g_{lm}, l, $m=0,1,\cdots,127$로 표시하고 g_{lm}는 "X"나 혹은 공란(blank)이 되는데 만약 $X_i=(L_i, U_i)$의 1번째 비트가 $X_j=(L_j, U_j)$의 m번째 비트의 생성에 영향을 주면 "X"로 그렇지 않으면 공란이다.

G_{ij}를 다음 (1)식과 같이 4개의 부행렬로 나누어 각각 구함으로써 보다 효과적으로 얻어 낼 수 있다.

$$G_{ij}=\begin{vmatrix} G_{ij}^{(L,L)} & G_{ij}^{(L,U)} \\ G_{ij}^{(U,L)} & G_{ij}^{(U,U)} \end{vmatrix} \quad \cdots\cdots\cdots\cdots\cdots (1)$$

여기서 64×64 부행렬 $G_{ij}^{(L,U)}$는 U_i의 비트가 갖는 L_j의 비트에 대한 의존상태를 나타낸 것이다. 우선 인접 라운드간의 $G_{i,\,i-1}$를 먼저 구한 다음 이들을 이용하여 G_{ij}, i>j를 구하도록 한다. 그림 3으로부

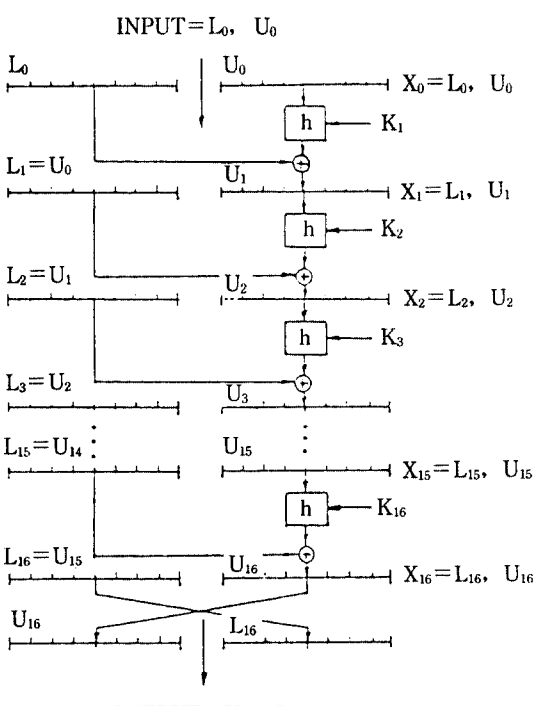

Fig. 3. Block Diagram of Lucifer Encryption Process.

터

$$L_i = U_{i-1} \quad \cdots\cdots (2)$$

$$U_i = h(U_{i-1},\ K_i)\ L_{i-1} \quad \cdots\cdots (3)$$

Table 3. Permutation for diffusion

8	17	50	59	28	5	46	39
16	25	58	3	36	13	54	47
24	33	2	11	14	21	62	55
32	41	10	19	52	29	6	63
40	49	18	27	60	37	14	7
48	57	26	35	4	45	22	15
56	1	34	43	12	53	30	23
0	9	42	51	20	61	38	31

로 둘 수 있다. 이들 식에서 $G_{i,\ i-1}{}^{(U,U)}$를 제외한 부행렬들을 그림 5와 같이 쉽게 얻을 수 있다. $G_{i,\ i-1}{}^{(U,U)}$를 얻기 위하여 U_{i-1}에 대한 표 3의 확산치환 직선의 출력 U_i 의 의존상태 행렬을 먼저 구한다. 그림 4에서 각 바이트에서만 대체와 치환이 수행됨을 알 수 있으므로, 그림 6과 같이 U_i 와 U_{i-1}간의 의존상태를 나타낼 수 있다. 그림 6에서 주의할 사항은 "X"로 표시된 8×8 소행렬이 실제로는 열쇠의 상호교환 제어 비트가 "0"이면 그림 7의 (a)의 형태로, "1"이면 (b)의 형태로 된다. 이 제어비트의 확률을 $p(0)=p(1)=1/2$로 가정하면 (a)와(b)가 나타날 확률이 1/2이 된다.

그림 6에 표 3의 확산치환을 적용시키면 U_i와 U_{i-1} 간의 의존상태를 얻을 수 있는데, 그 결과는 그림 8과 같이 됨을 알 수 있다.

다음으로는 $G_{i,\ i-1}$을 이용하여 $G_{i,\ j}$를 구하는 방법에 관해 살펴 본다. 관계식 $L_{i+1}=U_i$로부터 $G_{i+1,\ i-1}{}^{(L,L)}=G_{i,\ i-1}{}^{(U,L)}$ 및 $G_{i+1,\ i-1}{}^{(L,U)}=G_{i,\ i-1}{}^{(U,U)}$임을 알 수 있다. 즉, $G_{i+1,\ i-1}$의 상반부 64×128 행렬은 $G_{i,\ i-1}$의 하반부 64×128 행렬과 동일하다. 또한, 그림

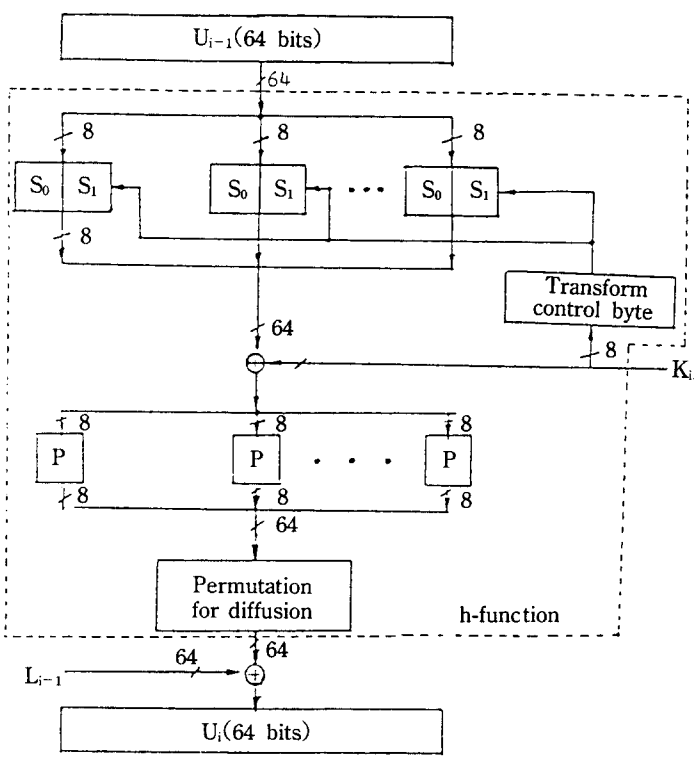

Fig. 4. Structure of the h-Function

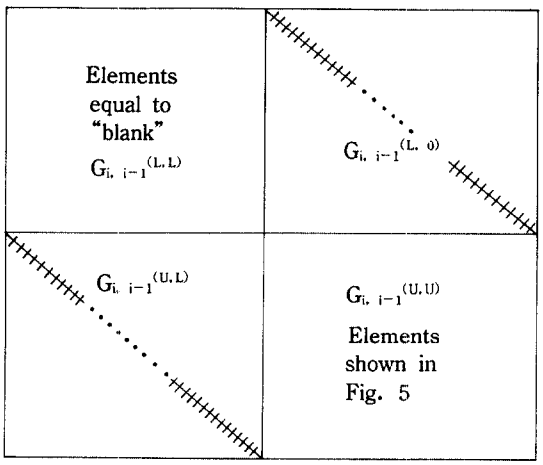

Fig. 5. Matrix $G_{i,\ i-1}$.

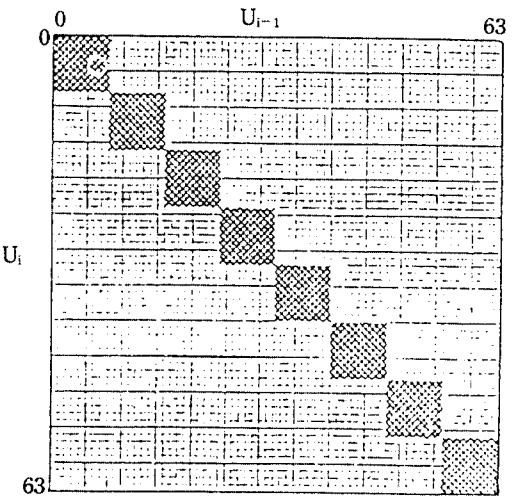

Fig. 6. Matrix $G_{i,\ i-1}^{(U,U)}$ Before the Permutation

3에서 볼 수 있듯이 U_{i+1}과 L_{i-1}의 관계는 U_{i+1}과 U_i간의 관계와 등가이고, L_{i+1}과 U_{i-1}의 관계는 U_i와 U_{i-1}간의 즉, U_{i+1}와 U_i간의 관계와 등가이므로, $G_{i+1,\ i-1}^{(U,L)}=G_{i+1,\ i-1}^{(L,U)}=G_{i,\ i-1}^{(U,U)}$이다. 따라서, $G_{i+1,\ i-1}$의 우측 상단 64×64 행렬과 좌측 하단 64×64 행렬은 서로 동일하다.

이제 나머지 부분 행렬 $G_{i+1,\ i-1}^{(U,U)}$를 구해본다. U_{i+1}의 각 비트(구체적으로 S번째 비트로 두고 설명함)는 그림 8에서 "X"로 표시된 U_i의 8개의 비트에 의존된다. 이들을 해석상 $m_i(s)_{i=1}$, i=1, 2, … 8로 둔다. U_i의 해당 $m_i(s)$ 비트는 같은 방법으로 U_{i-1}의 8비트에 의존되어 생성된다. 이를 나타내면 그림 9의 (a)이다. U_{i+1}과 U_i간의 의존상태를 나타낸 그림 8과 그림 9의 (a)로부터 U_{i+1}의 S번째 비트와 U_{i-1}간의 의존상태인 그림 9의 (b)를 얻을 수 있다. 이러한 과정을 U_{i+1}의 모든 비트에 적용시키면 그림 10의 $G_{i+1,\ i-1}^{(U,U)}$를 얻을 수 있다.

앞에서 구한 의존상태를 이용하여 i번째 라운드의 출력 $X_i=(L_i, U_i)$가 입력정보 비트 $X_0=(L_0, U_0)$에 대한 의존도를 $G_{i,\ 0}$ 행렬에서 "X" 표시의 점유 백분율로 나타낼 때, 라운드별 의존도는 표 4에 수록하였는데, 100% 의존도를 얻기 위해서는 5 라운드가 필요함을 알 수 있다. 5 라운드 이후에는 100% 의존도가 유지된다.

Table 4. Degree of Dependence(%) Between Ciphertext and Plaintext

Round	Degree of dependence
1	2.34
2	10.05
3	32.98
4	68.25
5	93.36
6	100.00

2) 키이와 암호문 심볼간의 상호 의존도

Lucifer에서 열쇠 비트가 i번째 라운드의 출력에 미치는 의존도에 대해서 128×128 행렬 F_i로 나타내어 조사한다. 행과 열의 (l, m)번째의 원소 $f_{l,\ m}$는 i번째 라운드의 출력 X_i의 1번째 비트의 생성에 열쇠의 m 비트가 영향을 미치면 "X"로, 그렇지 않으면 공란으로 둔다. (4)식과 같이 F_i를 분리하여 구성시키기로 한다.

$$F_i=\begin{bmatrix} F_i^{(L)} \\ F_i^{(U)} \end{bmatrix} \quad \cdots\cdots\cdots\cdots\cdots\cdots (4)$$

여기서, 부행렬 $F_i^{(L)}$는 X_i의 좌표 64비트 L_i에 128 비트의 열쇠가 미치는 의존상태를 나타내며, 64×

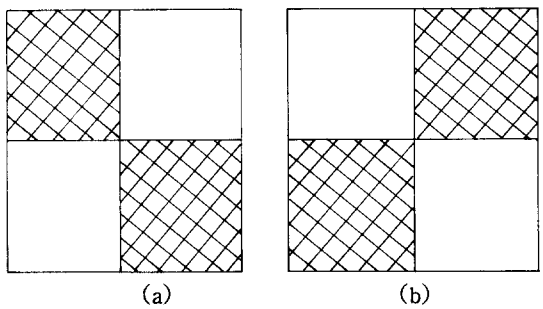

Fig. 7. Two Types of the "X" Marked Sub-matrix.

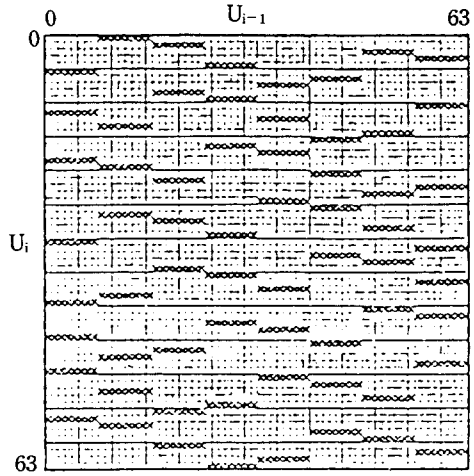

Fig. 8. Matrix $G_{i,\ i-1}^{(U,U)}$

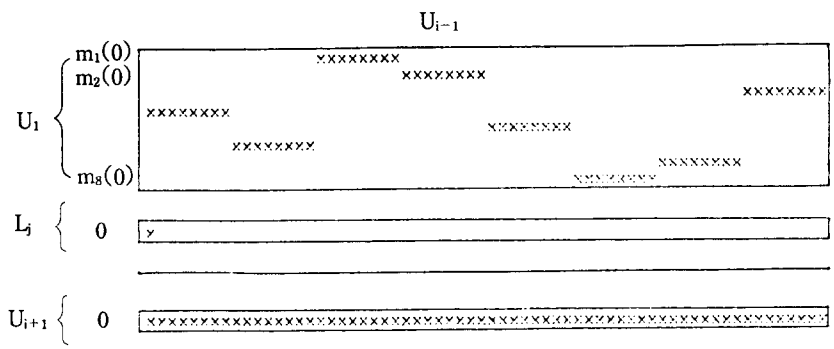

Fig. 9. Example of s=0 for $G_{i+1,\ i-1}^{(U,U)}$

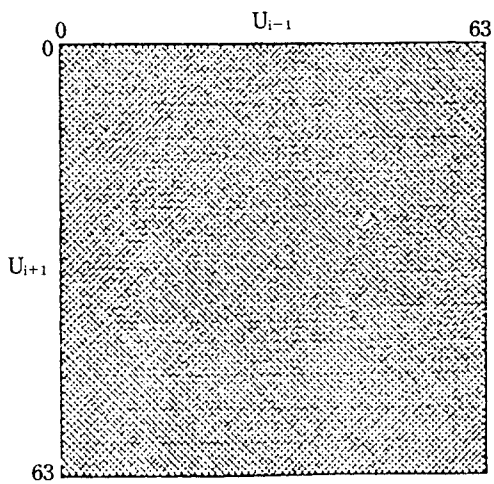

Fig. 10. Matrix $G_{i+1,\ i-1}^{(U,U)}$

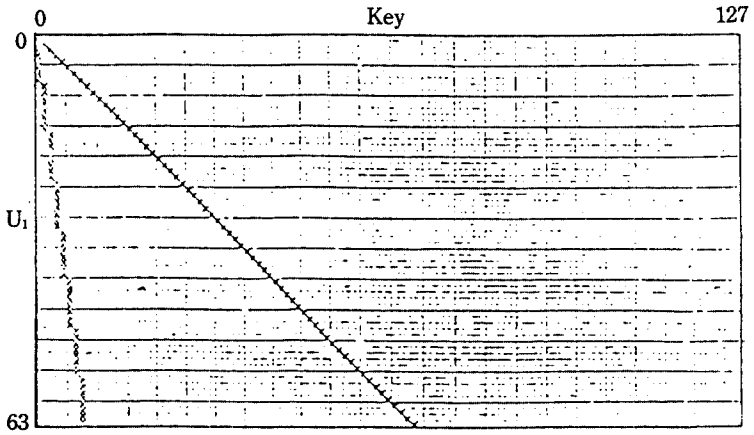

Fig. 11. Matrix $F_1^{(U)}$ Before the Permutation P

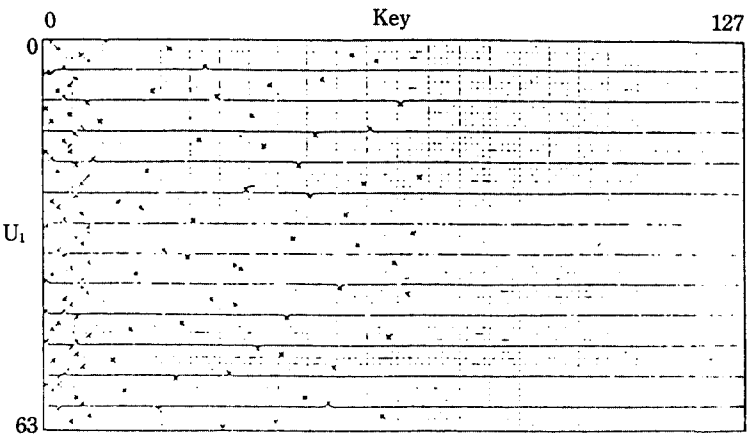

Fig. 12. Matrix $F_1^{(U)}$

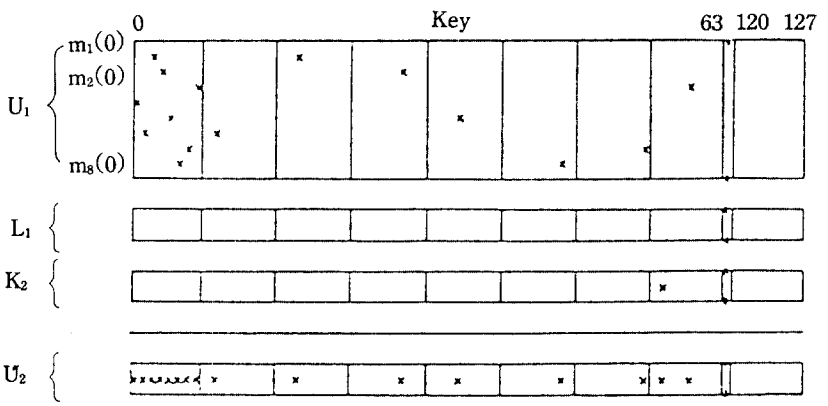

Fig. 13. Example of s=0 for $F_2^{(U)}$

128 행렬이 된다.

행렬 F_1를 먼저 구한 후에, 이것과 $G_{i,\ i-1}$로부터 모든 행렬 F_i를 구하기로 한다. 그림 3에서 $L_1(=U_0)$은 열쇠와는 무관하므로 $F_1^{(L)}$의 모든 원소는 공란이 된다. 그리고 $F_1^{(U)}$의 계산은 다음과 같다. 그림 4의 치환 직전의 출력 U_i과 열쇠간의 의존상태는 수행과정을 따라서 조사하면 그림 11의 행렬로 표시될 수 있음을 쉽게 알 수 있다. 그리고 치환과정 및 확산치환 과정을 적용시킨 최종적인 $F_1^{(U)}$ 행렬은 그림 12와 같다.

F_1과 $G_{i,\ i-1}$을 사용하여 $i>1$에 대한 F_i를 얻는 방법에 대해 살펴본다. 관계식 $L_2=U_1$으로부터 $F_2^{(L)}=F_1^{(U)}$임을 알 수 있다. 즉, F_2의 상단 64×128 부행렬은 F_1의 하단 64×128 부행렬과 같다. 따라서 $F_i^{(U)}=F_{i+1}^{(L)}$가 되므로 $F_i^{(U)}$를 얻으므로써 F_i를 구하게 된다. $F_2^{(U)}$의 임의의 한 행을 s행으로 두고 $s=0, 1, \cdots, 63$의 각 경우에 대해 알아본다. 앞에서 $G_{i+1,\ i-1}$를 구하는 방법과 같이 $F_1^{(L)}$의 $m_1(s)$에서 $m_s(s)$까지의 행들과 $F_1^{(L)}$의 s행을 결합하고 추가적으로 대체와 permutation for diffusion의 과정을 거쳐 나타나는 열쇠 K_2의 영향을 고려하여 $F_2^{(U)}$의 s행을 얻을 수 있다. 그림 13은 $s=0$인 경우의 예이며, $s=63$이 될때까지 이런 과정을 반복하면 $F_2^{(U)}$를 구하고 $F_i^{(U)}$, $i>2$도 같은 방법으로 얻을 수 있다. 표 5는 이상과 같이 구한 라운드별 열쇠에 대한 암어의 의존도를 백분율로 나타낸 것이다. 여기서 9번째 라운드에 이르러서야 열쇠의 의존이 100%가 됨을 알 수 있다. 그리고 열쇠의 전반부 63비트가 후반부 비트보다 암어에 미치는 영향이 강함을 알 수 있다.

Table 5. Degree of Dependence(%) Between Cipertext and Key

Round	Degree of dependence
1	0.78
2	4.53
3	16.87
4	46.47
5	77.89
6	99.53
7	98.75
8	99.76
9	100.00

3. 맺 음 말

그 구조에 있어서 Lucifer는 16 라운드 반복하는 형태이며, 매 라운드에서 고정된 대체와 치환을 수행한다. 대체에서 2개의 비선형적 대체 상자(S-box)를 사용하며, 치환과정에서는 8 비트 크기의 부블록으로 나누어 각 부블록내에서 치환을 수행한다. 대체와 치환이 고정된 것은 하드웨어용으로 설계된 것이며, 수행단위가 8 비트인 것은 개발당시의 마이크로프로세서의 기술수준과 관련이 있는 것으로 여겨진다. 평문과 암호문간의 상호 의존도에서 6 라운드에서부터 100%를 갖는다. 키이와 암호문간의 상호의존도에서는 9 라운드부터 100%의 의존도를 갖는다. 이러한 Lucifer의 심볼간 상호의존성은 DES나 FEAL-8에 비하면 약한 편이다.

참 고 문 헌

1. A. Sorkin, "Lucifer, a Cryptographic Algorithm," Cryptologia, Vol. 8, No. 1, pp. 22-35, Jan. 1984.

2. National Bureau of Standards, Data Encryption Standard, *U.S. FIPS PUB 46*, pp. 1-18, 1977.

3. A. Shimizu and S. Miyaguchi, "Fast Data Encipherment Algorithm FEAL," Abstract of Eurocrypt 87.

4. T. Cusick and M. Wood, "The REDOC-II Cryptosystem," Crypto'90, pp. 519-544, 1990.

5. C. Meyer and S. Matyas, *Cryptography : A New Dimension in Computer Data Security*, New York, John Wiley Sons, 1982.

□ 著者紹介

文 相 在(正會員)

1948年 4月 20日生
서울大學校 工業敎育科(電子專攻 學士)
서울大學校 大學院 電子工學科(碩士)
美國 UCLA 工學博士(通信工學 專攻)
金星電氣株式會社 勤務
美國 UCLA 硏究助員 勤務
美國 Satellite Tech. Management Inc. 勤務/美國 UCLA Postdoctor 勤務(Dept. of Elec. Eng.)/美國 OMNET 株式會社 Consultant 勤務
慶北大學校 電子工學科 副敎授
主關心分野： 符號技術 및 디지틀通信 등