□研究論文□

韓國 軍事運營分析 學會誌
第 15 卷, 第 2 號, 1989. 12. 31.

키에 의존하는 S-Box를 이용한 DES 소프트웨어의 개발 (Development of DES using Key-Dependent S-Boxes)

김 세 헌* 엄 봉 식*

요 약

여러 암호화 방식 중 비교적 비도가 높으며 표준화되어 있어 가장 널리 사용되고 있는 것이 DES이다. 그러나 DES의 안전성에 대하여 많은 문제점과 의문이 제기되어 왔으며 이들은 주로 DES S-Box에 관한 것이다. 이 S-Box의 구성은 평문과 암호문 사이의 상호관련성을 제거해주는 데 도움을 주지만 여기에 비밀통로가 숨겨져 있을 수 있는 가능성이 문제되고 있다. DES의 S-Box에 대한 이러한 문제점을 해결할 수 있는 방법의 하나는 0부터 15까지의 수를 무작위하게 순열하여 S-Box를 구성하는 것이다.

본 연구에서는 키에 의존하는 S-Box를 설계하며 아울러 이 변형된 DES를 수행할 수 있는 소프트웨어를 구축하여 변형된 암호화 방법의 안전성을 검토하고자 한다.

1. 서 론

정보에 대한 보호대책으로 가장 보편적이며 필수적인 방식은 정보를 암호화하는 것이다. 이러한 암호화 방식 중 비교적 비도가 높으며 표준화되어 있어 가장 널리 사용되고 있는 것이 DES(Data Encryption Standard)이다. DES는 IBM사의 Lucifer 암호[7]가 그 원형으로서, 1977년 미국상무성 표준국(NBS)에 의해 채택된 암호표준이다. NSA(국가 안전국)가

* 한국과학기술원

이 알고리즘을 검토했고 설계까지 영향을 미쳤다. 공표 당시부터 안전성에 대한 상당한 논란이 있었으며 그후 몇년이 지나자 이 논란은 몇가지 합의에 이르렀다[1]. 즉 NSA가 IBM의 원형에서는 128비트였던 키를 56비트로 줄인 것은 확실히 DES를 약화시켰으며 알고리즘에는 숨겨진 비밀통로가 있을 수도 있다. 그리고 DES의 안전성은 오래가지 못하리라는 것이다. 본 논문에서는 이러한 문제점들을 살펴보고 그에 대한 대응방안을 제시하고자 한다. 특히 키에 의존하는 S-Box를 설계하며 아울러 이 변형된 DES를 수행할 수 있는 소프트웨어를 구현하여 변형된 암호화 알고리즘의 안전성을 검토하고자 한다.

2. DES 알고리즘

전자통신망에서의 자료에 대한 안전성과 기밀성이 크게 요구됨에 따라 표준화된 암호화 알고리즘의 필요성이 제기되었다. 이에 알고리즘이 공표되어 암호화의 기밀성이 사용되는 키에 의존하며, 전자장치로 용이하게 구현되어 효율적으로 암호화할 수 있어 광범위하게 이용될 수 있는 암호화 알고리즘을 미국상무성 표준국(NBS)이 공모하였는데 이에서 채택된 암호화 방식이 Lucifer암호를 발전시킨 IBM사의 DES이다. 이는 NBS에서 1977년 미국정부의 공식 암호화 방식으로 발표되었다.

DES는 56비트의 키의 통제하에 64비트의 평문블럭을 64비트의 암호문으로 암호화하는 블럭암호화 방식이다. 입력되는 64비트의 키 중에서 각 바이트의 마지막 비트는 홀수 패러티비트로 이용하고 나머지 56비트가 유효한 키의 크기가 된다. DES의 암호화 과정은 키에 의해 통제되는 치환연산(substitution)과 고정된 순열연산(permutation)으로 구성되는 16개의 라운드로 구성된다. 〈그림 1〉은 치환연산을 수행하는 비도함수 f에 의하여 수행되는 암호화 과정을 개략적으로 보여주고 있다. 라운드 i에서는 키발생 방식에 의해 56비트의 암호키에서 생성된 48비트의 서부키(sub-key) K_i를 사용한다. 암호화 과정의 첫단계에서는 주어진 평문 블럭을 〈그림 2〉의 초기순열표로 순열을 하고 16개의 라운드가 끝난 후에는 〈그림 3〉의 초기순열의 역순열을 하게 된다.

초기순열을 한 후에는 그 출력이 왼쪽 32비트 L_{i-1}와 오른쪽 32비트 R_{i-1}로 나누어져서, R_{i-1}이 다음 라운드의 왼쪽 비트들 L_i가 되고, 부키 K_i의 통제하에 R_{i-1}의 비도함수값과 L_{i-1}의 exclusive-OR가 오른쪽 비트들 R_i가 된다. 즉 각 라운드는 다음의 연산을 한다.

$$L_i = R_{i-1}$$

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

DES는 키에 의해 통제되는 입력과 출력이 64비트인 하나의 치환함수(S-Box)라 할 수 있다. 그러나 이러한 하나의 S-Box로 암호화 방식을 구성하는 것은 비현실적이므로, DES

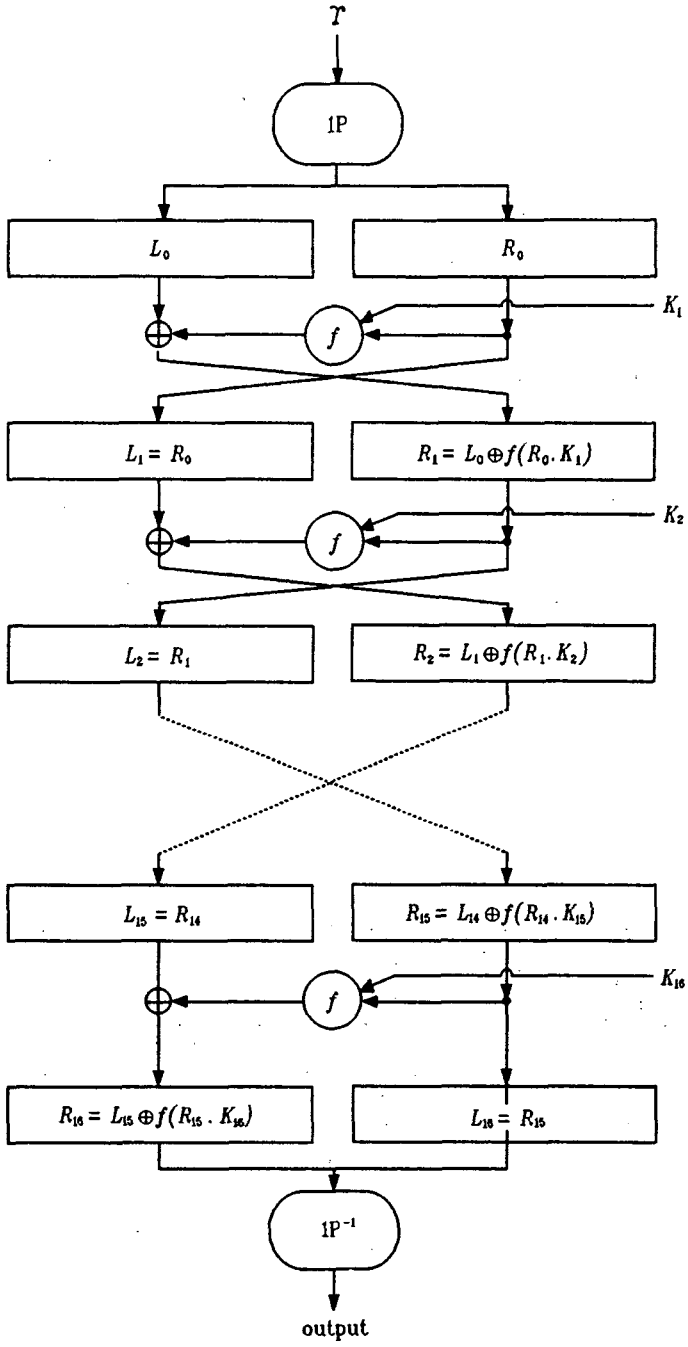

그림 1 DES 암호와 알고리즘

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

그림 2 초기 순열 P

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

그림 3 최종 순열 P^{-1}

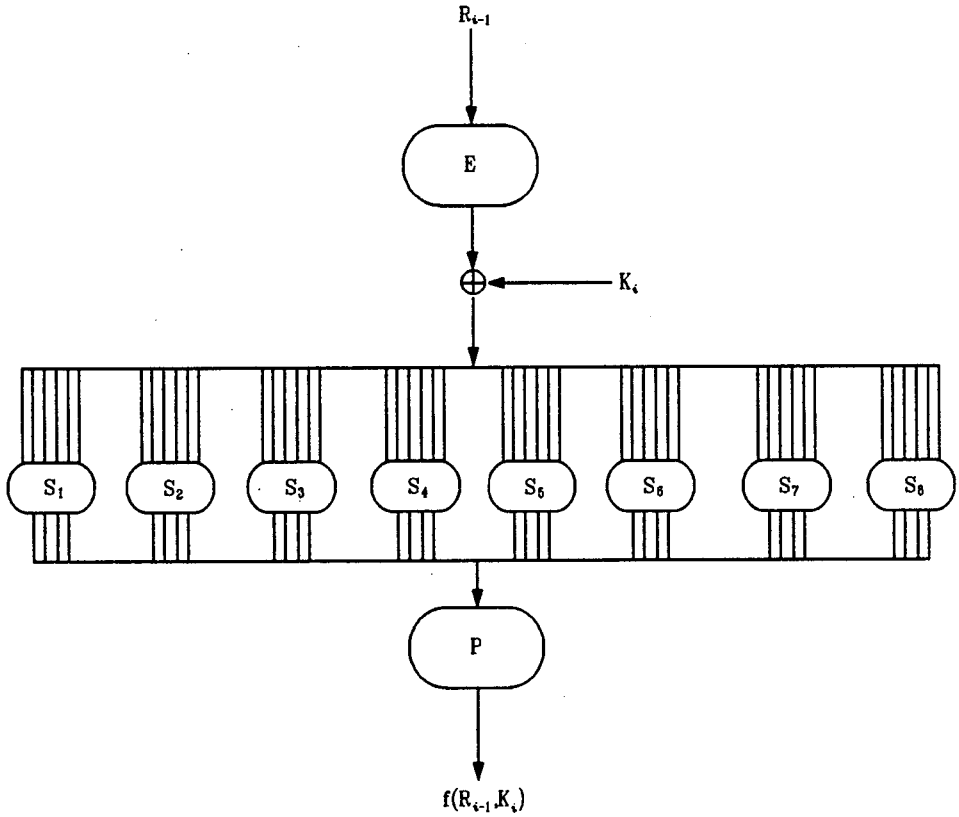

그림 4 비도 함수

는 비도함수내에 여러개의 작은 S-Box들을 이용하여 치환연산을 한다. 이러한 치환과 순열과정을 여러번 반복함으로써 DES는 암호학적 안전성을 높인다.

DES의 i번째 라운드에서의 비도함수 $f(R_{i-1}, K_i)$는 전 단계의 오른쪽 32비트를 48비트로 〈그림 5〉의 확장 순열시킨후, 이를 i단계의 부키 K_i와 exclusive-OR 연산을 한후 8개의 6비트 블럭으로 나누어져 〈그림 7〉의 8개의 S-Box를 통과하여 다시 32비트로 되어 주어진 순열 〈그림 6〉을 하여 그 값이 결정된다. DES의 8개의 S-Box는 6비트의 입력을 받아들여 그 중 첫 비트와 마지막 비트는 적용되는 치환의 종류를 결정하고 나머지 4비트는 정해진 치환에 의해 일정한 값으로 대체시켜 준다. 치환의 종류를 선택하는 비트들은 키비트뿐만이 아니라 평문비트에 의해 결정되므로 확장 순열과 S-Box는 self-keying효과를 지닌다. 따라서 각 단계의 S-Box의 입력간에는 Hamiltonian cycle이 존재하여 입력의 변화를 신속하게 확산시켜 주는데 효과적이다. DES의 성격상 암호블럭의 각 비트들은 평문블럭과 암호화 키의 함수이다. DES에서는 암호블럭의 각 비트들이 5라운드만 거치면 평문블럭과 암호화 키의 영향을 받게 된다.

S-Box의 구성은 비선형성을 지니고 있어 암호분석에 대해 저항력이 크다. 즉 DES 내의 다른 모든 연산은 선형성을 가지므로 DES의 비선형성은 $f(R_{i-1}, K_i)$에 기인하며, 이는 S-Box의 비선형성에 크게 의존하므로 어느 면에서는 S-Box의 구성이 DES의 안전성을 보장해주고 있다. 그러나 S-Box내의 치환의 선택기준이 공표되지 않아서 이들이 단순한 무작위 순열인지 아니면 일정한 비밀통로를 만들기 위해 특별히 설계되었는지 여부가 논란이 되고 있다.

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

그림 5 확장 순열 E

16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

그림 6 순열 P

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S1	0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S2	0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S3	0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S4	0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S5	0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S6	0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S7	0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S8	0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
	1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

그림 7 DES의 S-Box 표

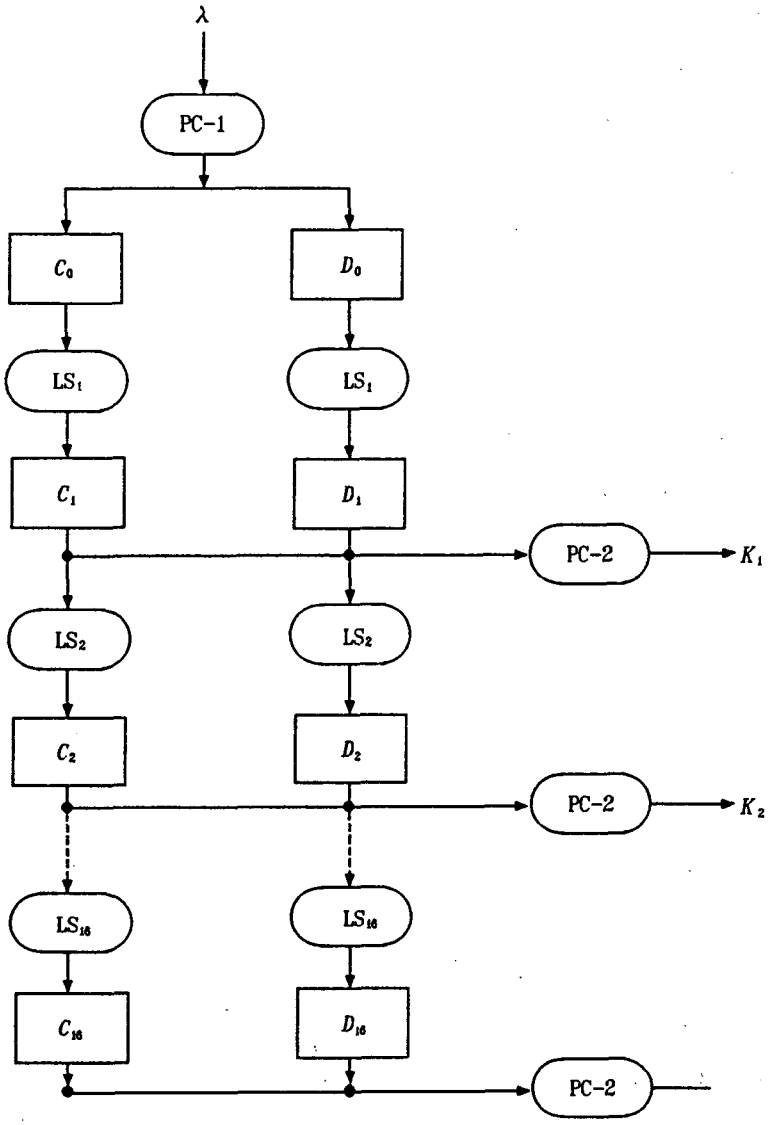
λ
PC-1
C_0
D_0
LS_1
LS_1
C_1
D_1
PC-2
K_1
LS_2
LS_2
C_2
D_2
PC-2
K_2
LS_{16}
LS_{16}
C_{16}
D_{16}
PC-2

그림 8 서부키의 계산

각 라운드에서 사용되는 부키는 주어진 순열연산과 비트 shift연산으로 이루어진 〈그림 8〉의 과정을 통해 암호화 키로 부터 얻어진다. 64비트의 암호화 키는 〈그림 9〉의 키 순열연산을 통해 8개의 패러티 비트가 제외되어 순열된다. 그 값은 각각 28비트로 이루어지는 왼쪽과 오른쪽 비트들 C_{i-1}, D_{i-1}로 나누어져 i번째 라운드에서는 각기 〈그림 10〉의 left shift연산을 거쳐 〈그림 11〉의 키 순열연산을 통하여 48비트의 부키 K_i가 얻어진다.

암호화의 역함수인 복호화 과정은 비도함수의 내용과 무관하게 이루어질 수 있다. 이는 사용되는 부키들을 역순으로 사용하는 동일한 암호화 과정을 통해 이루어진다.

i단계	left shifts의 수
1	1
2	1
3	2
4	2
5	2
6	2
7	2
8	2
9	1
10	2
11	2
12	2
13	2
14	2
15	2
16	1

그림 10 left shift(LS) 표

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

그림 9 키 순열 PC-1

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

그림 11 키 순열 PC-2

3. DES의 문제점

DES에 대해 제기되고 있는 문제점은 크게 두가지로 나눌 수 있다. 우선 DES의 키 크기가 현재나 가까운 미래의 기술 수준에 비추어 너무 작다는 것이다. 그리고 두번째로는 알고리즘의 내부구조에 숨어있을 수 있는 약점에 관한 것이다.

NSA와 IBM의 타협으로 128비트에서 56비트로 축소된 키는 DES를 크게 약화시켰다. 심지어는 외국 정부나 기업이 DES를 사용하는 경우를 대비하여 약화시키려했다는 주장도 있다[1]. Hellman[9]의 계산에 의하면 1977년의 기술수준으로도 이미 DES의 보안은 위협을 받았다.

DES에 대한 가장 단순한 침투방법은 cipher-plaintext attack의 가정하에 2^{56}개의 가능한 모든 키를 시행하여 사용된 키를 발견하는 방법이다(key exhaustive search [4][5]). Diffie와 Hellman은 이러한 key exhaustive search를 수행할 수 있는 특수용도의 기계를 제시하였다. 이 기계는 10^6개의 IC가 병렬처리되어 1 us당 하나의 키를 검사할 수 있으며 그 비용은 2천만 달러정도로 추정하였다. 이 기계를 이용하면 평균 12시간만에 사용된 키를 찾아낼 수 있으며, 5년간의 감가상각을 고려하면 해답마다 5천 달러의 비용이 든다. 그러나 발달하는 기술수준을 감안하면 10년후에는 개당 10달러가 될 것이라 예측했다. 따라서 현재의 DES의 키의 크기는 불충분하며 안전성을 확보하기 위해서는 키의 크기를 증가시켜야한다고 주장했다. NBS는 그의 계산에 반대하는 입장이었는데, 기계의 설계와 제어의 비용이 기계의 규모에 따라 증가되며, IC회로에 일정한 에러율이 존재하고 그 기계가 상당한 부피를 갖는 등의 이유를 들어 훨씬 비용이 높은 것으로 예측했다. 이후 NBS는 전문가들을 통해 정보기술에 대한 예측 워크샵을 열어 현실적인 비용의 기계는 1990년대에 가서야 등장할 것이라 발표했다[13].

Exhaustive search에서 컴퓨터의 기억용량은 더 필요하지만 키의 탐색에 필요한 시간을 줄이는 방법으로 Hellman의 time - memory trade-off이 있다[4][10]. 이는 chosen plain-ciphertext attack의 가정하에서 암호분석가가 미리 상당량의 계산을 해두면 table-lookup 방법을 취하여, N이 키의 갯수일때 $N^{2/3}$까지 계산량을 줄일 수 있다. 이 경우의 비용은 해답당 약100달러가 되며 미리 계산하는데 걸리는 시간은 삼년 가량이 소요될 것으로 추정하였다. 여기서 삼년이란 시간과 $N^{2/3}$의 계산량은 시간과 기억용량의 trade-off관계에서 유도된 최적치에 해당된다. 암호사용자가 이러한 위협을 방지할 수 있는 쉬운 방법이 있기는 하지만 키 크기의 증가는 이러한 분석에 대해서도 DES의 안전성을 높여준다.

현재의 슈퍼컴퓨터 및 인공지능 분야의 발

전은 56비트의 키를 가진 DES에 대한 점증하는 위협이 되고 있다[2]. 이러한 exhaustive search에 대하여 현재와 가까운 장래의 하드웨어와 정보기술의 발전수준에서 충분한 안전성을 보장할 수 있는 키의 크기를 추정해 볼 필요성이 있다. 아직까지는 DES가 안전한 것으로 알려져 있으며 때문에 DES의 키의 크기가 늘어나게 될 경우 안전성은 높아지겠지만 비용이 늘어나게 되므로 이것에도 trade-off관계가 존재한다.

DES의 내부구조에 대해서도 많은 문제점과 의혹이 제기되어 왔다. 즉 알고리즘 내에 DES를 쉽게 풀 수 있는 지름길이 있거나 비밀통로가 숨겨져 있지 않을까하는 점이다. DES에 비밀통로가 있다면 S-Box에 존재한 것이라는 것이 일반적인 인식이다. 지금의 상황은 아무도 이 비밀통로를 발견하지 못했거나, 누군가 발견은 했어도 발표는 하지않은 상태이다. DES의 공표 초기부터 NSA가 DES에 비밀통로를 심어 놓았다는 논쟁이 있었다. 키의 크기가 줄어든 것과 S-Box에 변화가 생긴 것은 같은 시기이다. NBS는 S-Box에 대한 설계원칙을 공개하지 않았는데 만약 이 원칙을 공개했다면 비밀통로에 대한 논쟁은 그렇게 크게 일어나지 않았을 것이다. 숨겨진 의도가 없다는 의견도 있고 하드웨어 설계를 위한 고려라는 의견도 있다[2]. DES에 비밀통로가 있는 지를 확인하려는 여러 시도가 있었으나 크게 성공한 경우는 없었다.

S-Box가 linear 또는 affine mapping일 경우 쉽게 깨어진다. DES의 원형인 Lucifer cipher의 경우 S-Box의 입력과 출력의 크기는 비선형성을 보장하기 위해 2보다 큰 수를 택했다. 이 크기를 늘릴수록 필요한 기억용량이 늘어나므로 적당하게 4를 선택한 것이다. DES의 경우 S-Box는 역사상이 필요없고, 6비트의 입력에 4비트의 출력으로 이루어져 있다. 물론 DES의 S-Box는 모두 affine이 아니다.

S-Box는 단 하나의 입력의 변화에도 둘 이상의 출력의 변화를 일으킨다. 이 성질은 error propagation이라하며 암호의 강도를 높여주는 역활을 한다. 주어진 평상문에 대해서 키와 암호문의 상관관계가 쉽게 발견되면 암호분석가는 찾고있는 키에 가까운 방향으로 탐색하는 것이 가능하게 되고 계산량을 줄일 수 있다. S-Box는 키와 암호문의 상관관계를 효과적으로 산개시킨다. 확장과 순열연산은 S-Box의 입력을 6개로 만들어주고 output error progagation을 빨리 확산시키는 역할도 한다[6].

비밀통로를 찾으려는 노력들에 의해 알고리즘의 약점들도 발견됐다. DES는 알고리즘에서 수행되는 두가지 종류의 exclusive-OR 연산으로 하여 다음과 같은 complementation성질을 갖는다.

$$y = f_k(x) \rightarrow \bar{y} = f_k(\bar{x})$$

이 성질을 이용하면 chosen plain-ciphertext attack에서 계산량을 절반으로 줄일 수 있다. 그러나 이는 DES의 안전성에 대한 심각한 위협은 되지 못한다[6][16]. 또한 알고리즘의 구조로 인해 weak key와 semi-weak key가 존재한다. weak key는 다음과 같으며 이들 키로 두번 암호화하면 평상문이 나온다.

0101010101010101
FEFEFEFEFEFEFEFE
1F1F1F1F0E0E0E0E
E0E0E0E0F0F0F0F0

semi-weak key는 다음과 같으며 이들 키의 쌍아래 두번 암호화하면 평상문이 된다.

E0FEE0FEF1FEF1FE와 FEE0FEE0FEF1FEF
1FFE1FFE0EFE0EFE와 FE1FFE1FFE0EFE0
01FE01FE01FE01FE와 FE01FE01FE01FE01
1FE01FE00EF10EF1와 E01FE01FF10EF10E
01E001E001F101F1와 E001E001F101F101
011F001F010E010E와 1F011F010E010E01

그러나 이러한 키의 존재도 DES에 대한 근본적인 위협은 되지 못한다[16].

4. DES의 개선 방향

앞장에서는 현재까지 DES에 대해 제기되고 있는 문제점들을 살펴보았다. 이러한 문제점에 대한 대안은 여러 가지 방향에서 고려될 수 있다.

우선 기존의 DES 알고리즘을 그대로 사용하되 그 운영에서 안전성을 제고하는 방법이 있다. 이를 위해서는 다음과 같은 사항들에 주의해야 한다[2]. 키의 선택에 있어 weak key나 semi-weak key를 사용하지 않아야 하며 알파벳으로 제한된 키를 피하고, 키를 자주 바꾸어야 하며, 알고리즘적으로 키를 선택하고, 키를 패스워드로 사용해서는 안된다. 또한 키의 배포에 세심한 주의를 기울여야 한다.

8바이트 수준에서 DES는 치환암호이기 때문에 고정된 구조를 가진 상업용 화일, 일기예보, 컴퓨터 프로그램 등의 텍스트 화일은 주의를 요한다. 이들이 암호화되었을 경우 반복되는 부분이 나타나 암호분석가의 일을 쉽게 만들 수 있다. 따라서 DES의 운용에서 Cipher Block Chaining이나 Cipher Feedback을 사용하여야 한다.

그러나 이러한 운영상의 개선만으로는 불충분하다. DES의 안전성에 대한 가장 큰 문제점의 하나는 불충분한 키의 크기로 인해 exhaustive search에 취약하다는 것이다. 이러한 문제점은 알고리즘의 내부 구조와는 무관하며 키의 크기와 직접적으로 연관된 문제이므로 키의 크기를 증가시킴으로서 해결될 수 있다.

키의 크기를 늘리는 방법중의 하나는 알고리즘을 변경시키지 않고 키를 달리하여 여러번 암호화하는 방법이 있다. Coppersmith에 의하면 'DES는 여러 암호분석에 견디어왔으며 현재로서 가장 효과적인 분석방법은 exhaustive search방법이다. 만약 이것이 위협이 된다면 삼중 암호화 방식을 사용하면 된다[3].' 삼중 암호화 방식의 경우 계산량은 meet in the middle attack때문에 $2^{56\times3}$이 아니라 $2^{56\times2}$에 비례한다. 이러한 다중 암호화 방식에서 제기되는 문제점은 과연 DES가 다중 암호화 방식에 close되어 있는가이다. 예로서 단순 치환암호는 close되어 있어서 두번 암호화를 시행해도 한번 한 것과 똑같은 결과가 된다[5][9]. 아무도 DES가 close되어있음을 증명하지 못했다. 이를 증명하려는 이론적인 시도들과 통계적인 시도들이 있으며[8][14] 이들 연구결과는 DES가 closed가 아니라는 믿음을 뒷받침하고 있다.

그러나 다중 암호화 방식은 알고리즘 운영상의 개선으로 볼 수 있으며, 알고리즘 자체가 128비트이나 256비트 등의 키를 사용하게 설계하는 것이 안전성이란 측면에서 보다 바람직하다.

키 확장 알고리즘을 이용하여 키의 크기를 변하게 하는 방법도 있다. 128비트의 linear feedback shift register를 이용하면 원래의 키의 크기가 얼마이든지 128비트의 키를 갖게 된다[5].

암호화되는 블럭의 크기를 증가시키면서 키의 크기를 56비트에서 128비트이나 256비트로 늘리는 방법도 고려될 수 있다. 이때는 알고리즘 내부구조가 많이 달라져야 하며 암호의 강도를 보장하는 설계원칙이 제시되어야 한다[11]. 그러나 키의 크기가 증가하면 키의 관리에서 문제가 발생한다. 키의 크기를 128비트나 256비트로 늘려서 얻어지는 안전성의 제고가 키의 저장이나 배포 등에서 발생되는 문제를 능가하지 못하는 경우에는 8개의 패러티 비트를 모두 포함하여 64개의 키 비트를 사용하는 방법이 있다. 이는 가능한 키의 수를 256배 증가시킨다.

키의 크기를 증가시키는 것은 비밀통로에 의한 short cut search를 방지하는데도 유용하다는 사실을 고려할 때 바람직한 일이다. 여러 방안중 비용과 안전의 관계를 고려하여 최적안을 선택해야 한다.

그러나 S-Box에 비밀통로가 있는 경우에는 이와 같이 단순히 키의 크기를 증가시키는 것만으로는 불충분하며 알고리즘 구조 자체에 대한 수정이 요구된다. 본 요구에서는 S-Box 내의 순열을 키에 의존하여 무작위로 발생시킴으로써 키의 크기를 쉽게 증가시키고 또한 비밀통로를 회피할 수 있는 방법을 사용하고자 한다. 이에 대한 자세한 내용은 다음 절에서 다루고 있다.

5. 키에 의존하는 S-Box의 구성

우리는 S-Box내의 순열을 키에 의하여 발생되는 무작위 순열로 구성하고자 한다. 만약 DES에 비밀통로가 숨겨져 있다고 하더라도 이는 S-Box내의 순열에 크게 의존할 것이므로, 이와 같이 S-Box 자체가 변한다고 하면 그 비밀통로는 없어져 버릴 것이다. 다음의 방법에 의해 무작위 순열을 구할 수 있다[15].

```
Algorithm : Random Permutation
            Generator
  for i := 1 to n do
      A[i] := i;
  for i := n down to 2 do
    begin
      j := Trunc(Random × (i-1)) + 1;
      Exchange(A[i], A[j]);
    end;
```

여기서 키의 일부가 무작위 순열을 발생시키기 위한 난수의 초기치로 주어지게 된다. 우리는 이와 같은 방법으로 32개의 무작위 순열을 구하여 S-Box를 구성하게 된다.

이와 같이 키에 의존하는 S-Box를 사용하는 경우, S-Box를 선택하는 키는 기존의 56비트와 추가적인 비트들로 이루어질 수 있으므로 쉽게 키의 확장이 가능하다. 가능한 모든 S-Box의 수는 $(16!)^{32} \approx 2^{1416}$이 되므로 난수 발생의 초기치로 사용되는 키는 1416비트까지 의미가 있을 것이다.

우리는 키에 의존하여 S-Box를 선택함으로써 S-Box의 구조자체를 변경하게 된다. 따라서 DES에 숨겨져 있을 수 있는 비밀통로를 파괴할 수 있으며, 알고리즘의 구조가 키에 보다 의존하므로 암호분석을 보다 어렵게 할 수 있는 것이다. 또한 효율적인 exhaustive search로 문제가 되는 키의 크기를 확장시킬 수 있다. 이 변형된 DES에서 S-Box의 구성은 키가 입력되어 각 단계의 부키를 발생시킬 때 한번 발생시켜서 이를 저장하고 사용하면 되므로 알고리즘을 소프트웨어적으로 구현하는 데에는 효율성에서 큰 문제가 되지 않을 것이다.

무작위적으로 선택된 S-Box가 선형이나 affine mapping이 되어 안전성이 위협받는 weak key가 존재한다. 따라서 이러한 weak key를 선택하지 않는 적절한 키의 선택 절차가 요구된다. 이는 키를 선택할 때 간단한 절차에 의하여 각 S-Box가 affine인지 확인할 수 있으며, S-Box의 출력이 3비트인 경우 affine하게 되는 확률을 3%정도 밖에 되지 않으며 DES와 같이 S-Box의 출력이 4비트인 경우 affine하게 되는 확률을 3%정도 밖에 되지 않으며 DES와 같이 S-Box의 출력이 4비트인 경우에 그 확률은 0.5×10^{-7}보다 작다. 따라서 무작위적으로 생성된 S-Box들이 변형된 DES 알고리즘을 affine하게 하는 경우는 거의 존

재하지 않을 것이다. 또한 S-Box들이 affine 이외에 어느 특정한 구조를 갖는 경우에 암호분석 측면에서 암호화 알고리즘이 취약해진다고 하면, 이 역시 키 선택과정을 통해 그러한 키를 배제할 수 있다.

알고리즘 자체가 키에 의존하여, 키에 따라 그 구조가 변하게 되므로 소프트웨어상의 구현에서는 문제가 없지만 하드웨어적인 구현에서는 알고리즘의 효율성이 크게 저해될 수 있다. 마지막으로 이 변형된 DES의 안전성에 대한 암호분석적인 충분한 검토가 이루어져야 하겠다. 즉 역사적으로 보아서 암호화 알고리즘의 설계상의 고려와 암호분석은 무관한 경우가 많았다는데 유의할 필요가 있다.

6. 알고리즘의 구현

앞에서와 같이 설계된 변형된 DES를 소프트웨어적으로 구현하였다. 무작위 순열을 발생하는데 사용되는 난수발생기법은 몇가지 기준을 만족시켜야 한다. 먼저 난수발생기법의 효율성은 그리 중요한 문제가 되지 않는다. 이는 난수발생기법은 S-Box의 순열 값을 결정하는 암호화 방식의 초기화에서만 사용되기 때문이다. 난수발생기법은 충분한 크기의 주기를 가져야 한다. 주기가 작으면 선택될 수 있는 S-Box의 종류가 크게 제한되며 이는 키의 확장효과를 잠식하는 기능을 할 뿐아니라 가능한 S-Box의 종류가 제한됨으로써 암호분석 측면에서 취약해 질 것이다. 난수는 무작위성을 갖는 것이 중요하다. 이는 발생된 S-Box의 순열값이 명백한 구조를 가지는 것을 방지하여 주며 변형된 DES에 대한 암호분석을 어렵게 해줄 것이다. 마지막으로 발생된 난수들이 암호분석적 입장에서 안전해야 한다. 즉 S-Box의 일부 순열값이 알려진 경우에 이로부터 전체 순열값을 복원하는 것이 불가능하여야한다.

우리는 구현된 알고리즘에서 multiplicative linear congruence generator를 무작위 순열을 발생하는데 사용하였다. 이 방법은 단순하여 효율적이고 충분한 주기를 지니며 상당한 정도로 무작위성을 띠기는 하지만 이는 몇 개의 난수를 알고 있으면 쉽게 필요한 매개변수를 추정할 수 있다는 단점이 있다. 그러나 S-Box의 순열값이 일부 노출되어도 이는 난수값 자체가 노출된 것이 아니기 때문에 난수가 가지는 값의 크기가 충분히 큰 경우에는 큰 문제 없이 사용 가능하다. 사용된 매개변수는 다음과 같다.

$$X_i = aX_{i-1} \bmod m$$

$$a = 2^{31}-1$$

$$m = 7^5$$

그외에 사용할 수 있는 난수발생기법으로 quadratic residue generator와 같은 암호분석에 대하여 안전한 난수발생기법들이 고려될 수 있다.

본 구현에서는 알고리즘의 각 단계에서 비도함수내의 exclusive-OR 연산을 위한 키(비도키)와 S-Box를 발생시키기 위해 초기치로 사용되는 키(순열키)를 구분하였다. 즉 비도키는 원 DES와 같이 64비트를 받아들여 각 바이트의 마지막 비트는 패러티비트로 사용하고 나머지 56비트를 이용하여 비도함수내의 부키들을 발생시킨다. 그리고 비도키와는 별도의 32비트를 받아들여 이 역시 각 바이트의 마지막 비트는 패러티 비트로 사용하고 나머지 28비트를 난수발생의 초기치로 이용하였다. 이와 같이 비도키와 순열키를 구분하는 것은 이들 키간에 독립성을 유지하기 위해서이다. 이들 키가 중복되어 이용되는 경우 어느 하나가 노출되는 경우 나머지 키가 속할 수 있는 영역을 축소시킬 수 있을 뿐아니라 양 키의 상관관계에 의하여 암호분석에 대하여 많은 구조적인 약점을 제공하게 될 것이기 때문이다.

'SBox'라는 비트의 키를 입력하여 multiplicative linear congruence generator의 초기치로 이용하여, 무작위 순열발생 알고리즘에 의하여 발생된 S-Box는 〈그림 12〉와 같다. 비도키가 'Security'인 경우에 위에서 발생된 키를 이용하여 〈그림 13〉의 평문을 암호화하면 그 복호문은 〈그림 14〉와 같다.

변형된 알고리즘의 소프트웨어상의 구현에서는 단순히 난수발생기법과 무작위 순열발생 알고리즘의 도입만 추가적으로 필요하므로 그 구현이 원래의 DES보다 크게 복잡해지지 않는다. 이와 더불어 무작위 순열키를 구성하기 위한 계산량은 아주 적고 또한 이는 암호화 단계 초기에 한번만 수행하면 되므로 그 효율성은 초기의 극히 적은 초기화 단계를 제외하고는 DES와 동일하다.

우리는 초기의 적은 추가적 계산과 작은 추가적 기억용량으로써 기존의 DES내의 비밀통로를 방지하며, 아울러 키에 따라 알고리즘의 구조가 변함으로써 DES와 같이 공개된 암호화 알고리즘에서와 같은 집중적인 구조 분석에 대해서도 보다 높은 비도를 유지할 수 있다.

7. 결 론

본 연구에서는 DES가 지니는 여러가지 문제점과 쟁점에 대하여 논하였으며 이에 대한 여러가지 대책을 살펴 보았다. 특히 S-Box가 지닐 수 있는 약점을 극복하고 키의 크기도 확장시킬 수 있는, 키에 의존하는 S-Box를 고찰하고 이를 소프트웨어적으로 구현하고자 하였다. 이 변형된 DES의 연구와 개발은 보다 안전한 민간분야의 상업용 암호화 알고리즘의 개발에 기여하리라 여겨지며 이를 위해 변형된 DES의 안전성에 대한 보다 집중적인 연구가 요구된다.

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S1	0	1	11	14	8	7	0	10	15	4	6	5	3	13	9	12	2
	1	6	9	13	10	2	4	14	3	15	7	12	5	11	0	8	1
	2	12	0	4	9	5	14	7	15	13	1	11	8	2	3	6	10
	3	14	0	8	11	6	1	5	3	9	4	7	13	15	2	12	10
S2	0	14	10	3	8	12	1	4	9	6	11	13	2	0	15	5	7
	1	15	3	1	13	11	6	4	12	14	10	8	9	2	0	7	5
	2	2	11	6	14	3	9	13	1	7	10	15	5	0	8	12	4
	3	14	3	8	5	12	4	11	1	15	7	13	10	6	0	2	9
S3	0	4	10	9	2	5	14	7	0	15	11	13	6	1	3	8	12
	1	7	6	8	10	0	12	14	5	3	11	4	1	13	15	2	9
	2	10	11	13	9	14	12	7	0	2	8	1	4	15	5	3	6
	3	12	2	14	10	7	9	5	0	1	13	11	8	6	15	4	3
S4	0	4	5	12	9	8	14	11	13	1	15	2	7	0	10	3	6
	1	8	11	7	14	2	0	9	5	6	1	3	10	15	12	13	4
	2	6	9	10	7	13	0	2	5	1	15	11	12	8	14	3	4
	3	10	14	3	5	1	11	0	4	2	15	7	6	9	8	12	13
S5	0	9	3	5	13	2	11	4	1	7	8	15	10	14	12	6	0
	1	5	6	1	0	12	10	14	8	9	2	13	4	3	7	15	11
	2	12	11	14	10	13	7	4	6	3	8	1	15	9	2	0	5
	3	12	2	13	7	3	0	9	6	11	14	5	15	8	10	1	4
S6	0	14	10	11	12	3	4	1	13	0	15	8	5	6	9	7	2
	1	2	11	1	10	0	14	13	8	5	6	12	9	15	3	4	7
	2	3	11	5	1	7	8	0	9	10	13	12	15	6	2	4	14
	3	10	9	6	2	7	11	13	15	1	14	3	12	8	4	0	5
S7	0	3	15	14	12	9	10	0	2	1	13	6	8	11	7	5	4
	1	15	13	1	8	5	2	0	4	12	11	3	14	9	10	6	7
	2	13	3	14	10	1	11	0	2	12	6	8	15	7	5	9	4
	3	10	7	1	9	11	2	12	14	5	4	3	13	0	8	15	6
S8	0	5	12	7	8	0	15	9	13	1	2	6	14	4	11	3	10
	1	3	6	11	12	5	0	13	1	9	7	15	14	2	4	10	8
	2	6	8	15	13	12	10	11	5	4	3	9	7	0	14	2	1
	3	4	11	9	7	13	10	8	2	0	14	1	6	15	3	12	5

그림 12 무작위로 발생된 S-Box 표 (키 : SBox)

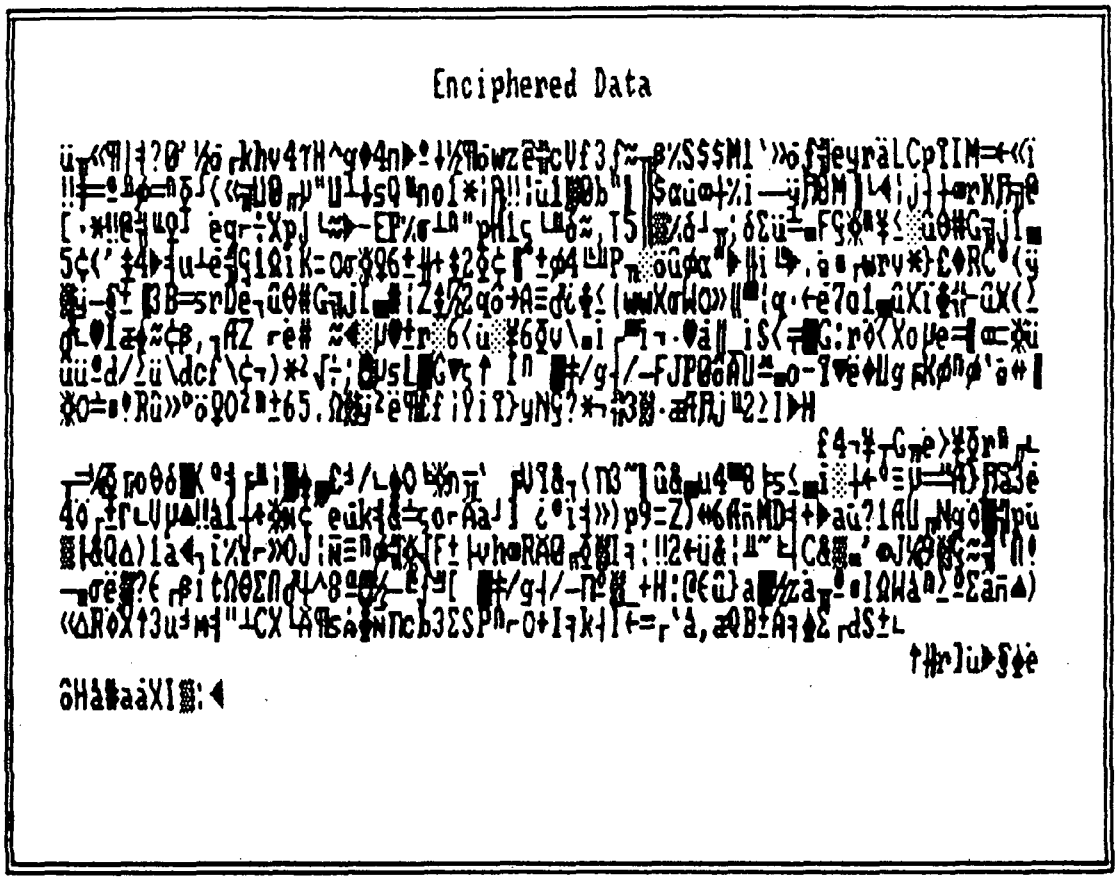

그림 13 키에 의존하는 S-Box를 사용한 DES의 암호문

Deciphered Data

Cryptography

Cryptography is the science of secret writing. A cipher is a secret method of writing, whereby plaintext is transformed into ciphertext. The process is called encipherment; the reverse process of transforming ciphertext into plaintext is called decipherment. Both encipherment and decipherment are controlled by a cryptographic key.

There are two basic types of ciphers: transpositions and substitutions. Transposition ciphers rearrange bits or characters in the data. With a "rail-fence" cipher, for example, the letters of a plaintext message are written down in a pattern resembling a rail fence, and then removed by rows.

Substitution ciphers replaces bits or characters, or blocks of characters with substitutes. A simple type of substitution cipher shifts each letter in the English alphabet forward by K positions ; K is the key to the cipher.

그림 14 키에 의존하는 S-Box를 사용한 DES의 복호문

참 고 문 헌

[1] T. Athanasiou, "DES Revisited," Datamation, Oct. 15, 1985, pp. 110-114.

[2] J. Carroll, "Strategies for Extending the Useful Lifetime of DES," Computers & Security 6, 1987, pp. 330-313.

[3] D. Coppersmith, "Cryptography," IBM J. Res. Develp. Vol. 31, No. 2, March, 1987.

[4] D. E. Denning, "Cryptography and Data Security," Addison-Wesley, Reading, MA, 1983.

[5] W. Diffie and M. Hellman, "Exhaustive Cryptanalysis of the NBS Data Encryption Standard," Computer, June, 1977, pp. 74-84.

[6] W. Diffie and M. Hellman, "Privacy and Authentication: An Introduction to Cryptography," Proc. IEEE, Vol. 67, No. 3, March, 1979.

[7] H. Feistel, "Cryptography and Computer Privacy," Scientific American, Vol. 228, No. 5, May, 1973, pp. 15-23.

[8] O. Goldreich and S. Even, "DES-Like Functions can Generate the Alternating Group," IEEE Trans. Inform. Theory, IT-29, 1983, pp. 863-865.

[9] M. Hellman, "DES will be totally insecure within ten years," IEEE spectrum, July, 1979.

[10] M. Hellman, "A Cryptoanalytic Time-Memory Trade off," IEEE Trans. Inform. Theory, Vol. IT-26, No. 4, July, 1980.

[11] J. Kam and G. Davida, "Structured Design of Substitution-Permutation Encryption Networks," IEEE Trans. Computers, Vol. c-28, No. 10, Oct., 1979.

[12] National Bureau of Standard, "Data Encryption Standard," NBS Publication NBSIR 76 1189, Dec., 1976.

[13] NBS, "1976 Workshop on Estimate of Significant Advances in Computer Technology-Executive Summary," NBSIR 76 1189, Page iii.

[14] R. Rivest, A. Sherman and B. Kaliski, "Is the Data Encryption Standard a Group?," Proc. of Workshop, Eurocrypto'85, Linz, Austria, 1985.

[15] S. Sattolo, "An Algorithm to Generate a Random Cyclic Permutation," Information Processing Letters, Vol. 22, 1986, pp. 315-317.

[16] R. Winternitz, "A Secure One-Way Hash Function Built from DES," Proc. of 1983 Symposium on Security and Privacy. Sponsored by IEEE.