通信情報保護學會論文誌
第 9 券 第 3 號, 1999. 09

DES에 기반한 조합형 한글 암호 알고리즘

김윤정*, 박근수**, 조유근*

An Encryption Algorithm Based on DES for Composition Hangul Syllables

Yoonjeong Kim*, Kunsoo Park**, Yookun Cho*

요 약

본 논문에서 제안하는 HEA(Hangul Encryption Algorithm)는 초성, 중성, 종성의 다차원 구조로 이루어진 한글의 특성을 이용하여, 조합형 한글 음절을 암호화하여 조합형 한글 음절을 생성한다. HEA에서는 암호화 결과가 모두 한글 터미널에 표시 가능한 한글이므로, 출력 내용이 표시 가능해야 하는 메일 시스템 등에 유용하게 사용될 수 있다. HEA는 DES에 기반하고 있으며, 키 전수 탐색, differential cryptanalysis, linear cryptanalysis에 대하여 DES와 유사한 안전도를 가지며, 암호문의 음소별 임의성, 평문-암호문 연쇄효과, 키-암호문 연쇄효과도 갖는다.

ABSTRACT

In this paper we present a Hangul Encryption Algorithm (HEA) which encrypts composition Hangul syllables into composition Hangul syllables, using the non-linear structure of Hangul. Since ciphertexts generated by HEA are displayable characters, HEA can be used in applications such as Privacy Enhanced Mail (PEM) where ciphertexts should be displayable characters. HEA is based on DES, and it can be shown that HEA is as safe as DES against the exhaustive key search, differential cryptanalysis and linear cryptanalysis. HEA also has randomness of phonemes of ciphertexts, and satisfies plaintext-ciphetext avalanche effect and key-ciphertext avalanche effect.

Keywords: encryption algorithm, Hangul composition code, DES (Data Encryption Standard), differential cryptanalysis, linear cryptanalysis

I. 서론

보안의 중요성이 증대되면서, 컴퓨터 통신망에서의 자료 전송에 암호화 기법이 많이 사용되고 있다. PEM(Privacy Enhanced Mail)[1,2,3] 등의 전자 메일 시스템의 경우, 메일의 몸체 부분은 반드시 모니터 화면에 표시 가능한 텍스트이어야 하는데, DES (Data Encryption Standard)[4,5,6] 등의 기존 암호 알고리즘의 수행 결과 생성된 출력은 제어 문자(control characters)를 포함하고 있으므

* 서울대학교 컴퓨터공학과 시스템 소프트웨어 연구실 (yjkim@ssrnet.snu.ac.kr) (cho@ssrnet.snu.ac.kr)
** 서울대학교 컴퓨터공학과 컴퓨터이론 및 암호학 연구실 (kpark@theory.snu.ac.kr)

로 이를 표시가능 문자로 바꾸어 주는 코드 변환이 필요하다. 이러한 코드 변환은 암호화에 있어서 별도의 단계를 수행해야 하는 문제 뿐 아니라, 표시 가능 문자로 바꾸어 준 화일이 원래 화일의 4/3 배로 크기가 커지는 단점을 갖는다. 본 논문에서 제안하는 한글 암호 알고리즘 HEA (Hangul Encryption Algorithm)는 PEM 등의 메일 전송 시스템에서 유용하게 사용될 수 있는 것으로[25], 조합형 한글 음절이 입력으로 주어질 때 이를 암호화하여 한글 터미날에 표시 가능한 조합형 한글을 생성한다.

HEA는 초성, 중성, 종성 또는 초성, 중성으로 이루어진 한글의 다차원성을 이용하여 구성되며, 전체적인 동작은 DES에 기반하고 있다[4,5,6]. DES는 현재 가장 널리 쓰이고 있는 비밀키 암호 알고리즘으로, 키의 전수 탐색 방법, differential cryptanalysis, linear cryptanalysis 외에는 안전하다고 알려져 있다. HEA는 키 전수 탐색면에서는 DES보다 시간 복잡도가 약간 높으며, differential cryptanalysis에 대하여는 확률이 높은 characteristic을 찾기가 어려워서, linear cryptanalysis에 대하여는 linear expression을 구성하기가 어려워 어느 정도 안전하다고 볼 수 있다. 또한, HEA가 생성한 암호문은 음소별 임의성을 가지며, 평문-암호문 연쇄효과, 키-암호문 연쇄효과를 만족함을 실험을 통하여 확인할 수 있었다.

본 논문은 다음과 같이 구성되어 있다. 먼저 II 장에서는 연구의 배경을 기술하며, 이어서 III 장에서 한글 암호 알고리즘 HEA의 세부 내용을 기술한다. HEA의 안전도 및 수행 성능을 분석한 내용이 IV 장, V 장에 각기 기술되며, 끝으로 결론 및 추후 연구 사항을 기술한다.

II. 연구 배경

1. 한글 코드 시스템

한글은 14 개의 자음과 10 개의 모음으로 이루어진다. 하나의 한글 음절은 초성, 중성, 종성으로 구성된다. 초성은 10 개의 기본 자음과 9 개의 이중 자음으로 구성되며,

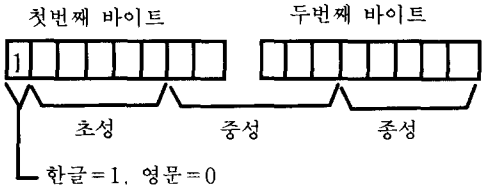


그림 1. 조합형 한글 코드
Fig 1. Composition hangul code

중성은 10 개의 기본 모음과 9 개의 이중 모음, 2 개의 삼중 모음으로 구성된다. 종성은 14 개의 기본 자음과 13 개의 이중 자음, 채움 문자로 구성된다. 이들이 표 1에 나타나 있다. 주로 사용되는 한글 코드 시스템은 조합형 코드 시스템(8 비트 2 바이트)과 완성형 코드 시스템(8비트 2 바이트: KSC 5601)이다[7,8]. 두 코드 시스템 모두, 한글 한 음절을 표시하는데 2 바이트를 사용한다. 그 외에, 한글을 지원하는 코드 시스템으로 유니코드 시스템이 있다[9].

조합형 코드 시스템은 한글 한 음소가 5 비트로 표시될 수 있다는 특성에 기반한다. 한글의 초성, 중성, 종성의 개수는 각각 19, 21, 28이므로, 5 비트로 표시될 수 있는 32보다 작다. 첫번째 바이트의 MSB (most significant bit)가 1인 경우, 그 바이트와 이어지는 1 바이트는 하나의 한글음절을 나타낸다. 첫번째 바이트의 MSB가 0인 경우, 그 바이트는 영문 ASCII 문자를 나타낸다. 한글을 나타내는 두 바이트의 비트 구성은 그림 1에 나타나 있다. 조합형 한글 코드 시스템은 현대 맞춤법에 따른 11,172 음절 글자를 모두 표시할 수 있다. 여기서 11,172는 19 (초성 갯수), 21 (중성 갯수), 28 (채움 문자 포함한 종성 갯수)을 곱하여서 얻어진 값이다.

완성형 코드 시스템 (KSC 5601)은 주로 쓰이는 한글 음절들을 선택하여 이들에 순서적으로 코드 값을 부여한다. 완성형에서 한 음절 표시에 사용되는 2 바이트는 각각 0xa1부터 0xfe까지의 값을 가질 수 있다. 즉, 한 바이트로 표시될 수 있는 $2^8 = 256$ 개의 코드 중에서 94개의 코드만을 사용한다. 이것은 ASCII 문자 및 확장 제어 문자와의 충돌을 막기 위해서이다. 한 바이트가 94개의

표 2. 한글 코드
Table 1. Hangul code

HEA 코드	초성	중성	종성
0	ㄱ (2)	ㅏ (3)	채움 (0)
1	ㄲ (3)	ㅐ (4)	ㄱ (2)
2	ㄴ (4)	ㅑ (5)	ㄲ (3)
3	ㄷ (5)	ㅒ (6)	ㄱㅅ (4)
4	ㄸ (6)	ㅓ (7)	ㄴ (5)
5	ㄹ (7)	ㅔ (10)	ㄴㅈ (6)
6	ㅁ (8)	ㅕ (11)	ㄴㅎ (7)
7	ㅂ (9)	ㅖ (12)	ㄷ (8)
8	ㅃ (10)	ㅗ (13)	ㄹ (9)
9	ㅅ (11)	ㅘ (14)	ㄹㄱ (10)
10	ㅆ (12)	ㅙ (15)	ㄹㅁ (11)
11	ㅇ (13)	ㅚ (18)	ㄹㅂ (12)
12	ㅈ (14)	ㅛ (19)	ㄹㅅ (13)
13	ㅉ (15)	ㅜ (20)	ㄹㅍ (14)
14	ㅊ (16)	ㅝ (21)	ㄹㅌ (15)
15	ㅋ (17)	ㅞ (22)	ㄹㅎ (16)
16	ㅌ (18)	ㅟ (26)	ㅁ (17)
17	ㅍ (19)	ㅠ (23)	ㅂ (19)
18	ㅎ (20)	ㅡ (27)	ㅂㅅ (20)
19		ㅢ (28)	ㅅ(21)
20		ㅣ (29)	ㅆ (22)
21			ㅇ (23)
22			ㅈ (24)
23			ㅊ (25)
24			ㅋ (26)
25			ㅌ (27)
26			ㅍ (28)
27			ㅎ (29)

* 각 음소 옆에 괄호로 둘러싸인 것은 조합형 코드 값임

코드만을 지원하므로, 94×94 = 8836 개의 코드만이 표시될 수 있다. 한글 2350 음절과 한자 및 제어 문자가 완성형 코드로 표시된다. 즉, 완성형 코드는 한글을 2350 개만 지원한다는 단점을 갖는다.

본 논문에서 제안하고 있는 알고리즘은 조합형 코드 시스템에 기반하며, 알고리즘에서 사용한 한글 음소의 코드 값이 표 1에 나타나 있다. 이 값들은 대응 음소 별로 차례로 0부터 일렬로 주어졌다. 예를 들면, 초성인 'ㄱ'은 조합형 한글 코드 값은 2 이나, 본 알고리즘에서 사용한 코드값은 0 이다.

2. 기존 한글 암호 알고리즘

한글로 이루어진 문장을 암호화하기 위하여 현재까지 사용한 방법은 한글의 특성을 고려하지 않고 기존의 암호 알고리즘을 그대로 이용하는 것이었다. 완성형 한글을 위한 압축과 결합된 암호 기법이 제안되기도 하였는데, 이것은 압축시에 한글의 사용 빈도수를 고려한 방법이다[10].

III. 한글 암호 알고리즘

제안하는 한글 암호 알고리즘 HEA는 한글 한 음절이 초성, 중성, 종성의 다차원 구조로 이루어졌음을 이용하여 조합형 한글을 암호화하여 조합형 한글을 생성하며, 전체 동작은 미국의 NIST(National Institute of Standards and Technology)에서 제안한 DES(Data Encryption Standard)에 기반하고 있다[4,5,6]. HEA의 전체적인 과정이 그림 2에 나타나 있는데, 그림 2에서 3 가지 줄로 구성된 사각형의 단위는 한글 음절로, 각 줄은 각각 초성, 중성, 종성을 나타낸다. 한 줄로 구성된 사각형의 단위는 키 스케줄 단계인 경우는 비트, 암호화 단계인 경우는 5 비트이다. 개괄적 내용은 다음과 같다.

- 입력: 한글 평문 64 음절, 키로 사용될 한글 7 음절 또는 영문 8 문자
- 출력: 한글 암호문 64 음절
- 키 스케줄 단계

- 한글 7 음절 또는 영문 8 문자를 키로 받아, 16 개의 48 비트를 생성해 낸다. 즉, 생성된 각 K_i (i=1,2,...,16)는 48 비트로 구성된다. 이 과정에서 사용되는 연산은 순열 (permutation), 좌 환형 쉬프트 (circular left shift), 순열적 선택 (permuted choice) 이다.

- 암호화 단계

- IP (Initial Permutation): 한글 평문 64 음절에 순열을 적용하여, 새로운 64 음절을 생성한다.
- 암호화 단계: IP의 결과인 64 음절을 입력으로 받아 새로운 64 음절을 생성한다. 생성 방법은 아래의 연산을 16번 반복하는 것이다. 64 음절을 이등분하여 왼쪽 32 음절과 오른쪽 32 음절로 나눈다. 오른쪽 32 음절과 키 K_i (i는 첫번째 반복에서는 1,

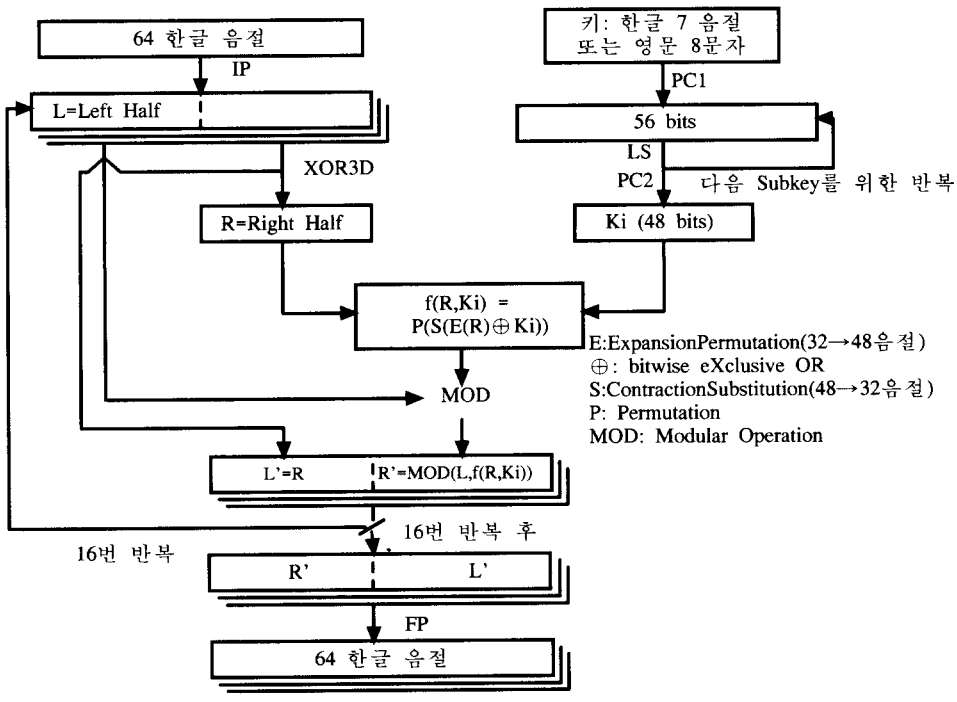


그림 2. 한글 암호 알고리즘
Fig 2. Hangul Encryption Algorithm

..., 열여섯번째 반복에서는 16이다)에 순열, 치환, exclusive-or 연산을 적용하여 32개의 5 비트를 생성한다. 이 결과를 왼쪽 32 음절에 모듈라 연산하여 새로운 32 음절을 생성한다. 이 32 음절을 오른쪽 32 음절로 하고, 원래의 오른쪽 32 음절을 왼쪽 32 음절로 하는 새로운 64 음절을 구성한다. 이 64 음절이 다음번 반복의 입력이 된다.

- FP (Final Permutation): 암호화 후 생성된 64 음절의 왼쪽 32 음절과 오른쪽 32 음절을 서로 치환하고, 이 결과에 IP의 역 순열을 적용하여 한글 64 음절을 생성한다.

1. 키 스케줄

영문 8 문자 또는 한글 7 음절이 PC1, LS, PC2 연산에 의하여 16 개의 48 비트로 변환된다.

1.1 PC1 (Permuted Choice1)

영문이 키로 주어진 경우 각 바이트의 하위 7 비트를 취하여 56 비트를 구성한다. 한글이 키로 주어진 경우 한글 한 음절로부터 초성과 중성 하위 4 비트를 취하여, 아래와 같이 총 56 비트를 구성한다. 종성은 채움 문자의 발생 빈도가 다른 음소들보다 상당히 높아, 키 구성에서 제외하였다.

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

1.2 LS (Left Shift)

PC1의 결과인 56 비트가 2 개의 28 비트로 분할되고, 각 28 비트에 왼쪽으로 환형 쉬프트가 수행된다. 16 번 반복 중 각 반복에서 쉬프트 하는 값은 DES와 같도록 구성하였는데, 각 반복 번호 i와 대응하는 환형 쉬프트 값 LS(i)는 아래와 같다.

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
LS(i)	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

1.3 PC2 (Permuted Choice2)

PC2는 56 비트로부터 48 비트를 선택하여 순열을 적용한 것이다. 이 규칙은 DES와 같도록 구성하였는데, 순열 값은 아래와 같다.

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 36 42 50 36 29 32

2. 암호화

한글 평문 64 음절에 키 스케쥴의 결과를 이용하여 암호화가 수행된다. 암호화는 처음과 마지막에 음절 단위로 수행되는 IP, FP 연산과, 16 번 반복되는 XOR3D, EP, XOR, S, P, MOD 연산으로 구성된다. XOR3D는 오른쪽 32 음절로부터 32 개의 5 비트를 생성하며, EP, XOR, S, P 연산은 이 32 개의 5 비트에 적용된다. MOD는 이 32 개의 5 비트와 왼쪽 32 음절로부터 새로운 32 음절을 생성한다. 그림 2에 f(R, K_i)로 표시되어 있는 것은 EP, XOR, S, P 연산을 말하며, 이것은 DES의 표준 입력 5 단위가 같은 키를 갖고 동시에 수행되는 것과 같다.

2.1 IP (Initial Permutation)

IP는 한글 64 음절에 처음으로 적용되는 순열이다. 순열 규칙은 DES와 같도록 구성하였는데, 값들은 아래와 같다. 이 규칙에 따라 음절 단위로 순열이 수행된다.

L0	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
R0	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

2.2 XOR3D (eXclusive-OR of 3 dimensions)

XOR3D는 32 음절로부터 32 개의 5 비트를 생성한다. 생성 규칙은 각 음절의 초성, 중성, 종성의 같은 비트 위치에 있는 3 가지 값을 비트 별로 exclusive-or하여 1 비트를 생성하는 것이다. 각 초성, 중성, 종성이 5 비트로 이루어지므로, 32 음절로부터 32 개의 5 비트가 얻어진다.

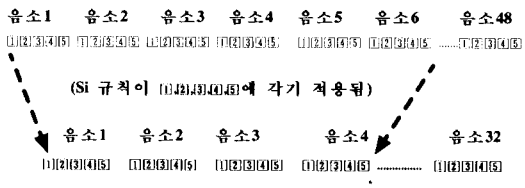


그림 3. S 규칙이 적용되는 방식
Fig 3. The way of adopting S rules

2.3 EP (Expansion Permutation)

EP는 32 개의 5 비트로부터 확장 순열을 적용하여 48 개의 5 비트를 생성한다. 이 순열 규칙은 DES와 같도록 구성되었다. 본 HEA의 EP가 DES의 EP와 다른 점은 순열이 한 비트가 아닌 5 비트 단위에 적용된다는 점이다.

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

2.4 XOR (eXclusive-OR)

XOR는 48 개의 5 비트와 키 48 비트로부터 새로운 48 개의 5 비트를 생성한다. i (i=1,2,...,48)번째 5 비트의 각 비트는 키의 i 번째 비트와 비트 별로 exclusive-or를 수행하여 새로운 i 번째 5 비트를 생성한다.

2.5 S (Contraction Substitution)

S 연산을 수행하여 입력 각 5 비트 중 i (i=1,...,5) 번째 비트로 이루어진 48 비트로부터 32 비트를 생성하는데, 이 32 비트는 출력 32 개 5 비트의 i 번째 비트를 구성한다. 이러한 S의 구성 원칙이 그림 3에 나타나 있다. 48 개의 i (i=1,...,5) 번째 비트로부터 32 개의 i 번째 비트를 구성하는 방법은 다음과 같다. i 번째 48 비트는 8 개의 6 비트로 나뉘어지고, 각 6 비트는 규칙 S_j (j=1,2,...,8)에 의하여 4 비트로 변환된다. 표 2에 규칙 S_j (j=1,2,...,8)가 나타나 있는데, 세부 규칙은 DES와 동일하다. 입력 6 비트 중 첫번째와 마지막 비트가 S_j의 행을 결정하고, 가운데 4 비트가 S_j의 열을 결정하며, 이 행과 열에 의해 지시된 위치에 있는 값이 출력 4 비트가 된다.

표 2. S 규칙
Table 2. S rules

S_1

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

S_2

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

S_3

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

S_4

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

S_5

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

S_6

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

S_7

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

S_8

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

2.6 P (Permutation)

P는 32 개의 5 비트에 순열을 적용한다. 순열의 적용 단위는 5 비트이다. 순열 값은 DES와 동일하며 이 값들은 아래와 같다.

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

2.7 MOD (Modular Operation)

MOD 연산은 암호화 과정에서는 모듈라 덧셈, 복호화 과정에서는 모듈라 뺄셈이 수행된다. 모듈라 덧셈은 $L+f(R,K_i) \bmod N$이며, 모듈라 뺄셈은 $L-f(R,K_i) \bmod N$이다. 이때, L은 32 음절이며, $f(R,K_i)$는 32 개의 5 비트이다. 32 음절의 초성, 중성, 종성 각각이 $f(R,K_i)$의 32 개의 5 비트와 독립적으로 계산된다. 초성인 경우 N=19, 중성인 경우 N=21, 종성인 경우 N=28이다.

2.8 FP (Final Permutation)

16 번의 암호화 단계 후 생성된 64 음절의 왼쪽 32 음절과 오른쪽 32 음절을 서로 치환하고 이 결과에 IP의 역과정을 수행한다. 즉, 각 음절을 원래의 위치로 보내주는 순열을 수행한다. 순열 규칙은 다음과 같다.

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

3. 복호화

복호화 연산은 2 가지 차이점을 제외하고는 암호화 연산과 동일하다. 하나는 모듈라 연산으로, 암호화 시에는 모듈라 덧셈이, 복호화시에는 L에서 $f(R,K_i)$를 감하는 모듈라 뺄셈이 수행된다. 다른 하나는 적용되는 키 순서로, 암호화시에는 16 개의 48 비트 키가 $K_1, K_2,...,K_{16}$의 순서로 사용되고, 복호화시에는 암호화에서 사용된 16 개의 48 비트 키가 $K_{16}, K_{15},...,K_1$의 순서로 사용된다.

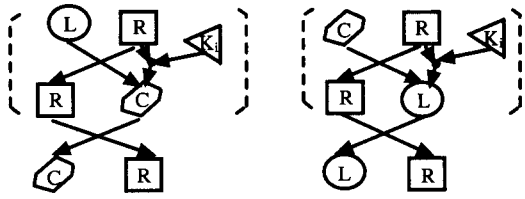


그림 4. 복호의 원리
Fig 4. The priciple of the decryption

복호의 원리가 그림 4에 나타나 있다. 암호화 시에, R은 그대로 L 부분이 되고 이것은 다시 R 부분이 된다. 복호화시에도 같은 과정을 거쳐 원래의 R이 얻어진다. L이 얻어지는 원리는 다음과 같다. 암호화시에 L, R, 키 K_i는 결과 $C = L + f(R,K_i) \bmod N$을 생성하고, 복호화시에 C, R, 키 K_i는 $L = C - f(R,K_i) \bmod N$을 얻어낸다. 즉, 평문에서 암호문이 얻어진 것 같이 복호 연산으로 암호문에서 평문이 얻어지는 것이다. 그림 4의 윗부분인 점선으로 둘러싸인 부분이 16 번 반복되어도 같은 결과가 얻어진다.

IV. 안전성 분석

암호 알고리즘이 안전하려면 기존의 알려진 침입 방법에 대하여 안전해야 하므로 [11], 우선 DES에 대한 기존 침입 방법인 키 전수 탐색 방법과 differential cryptanalysis 및 linear cryptanalysis에 대한 HEA의 안전성을 알아보았다. 이 외에, DES의 장점인 암호문의 임의성, 평문-암호문 연쇄효과, 키-암호문 연쇄효과를 HEA도 갖고 있는지 조사하였다.

1. 키 전수 탐색 방법

HEA는 키로 주어진 7 음절 중 각 초성과 중성의 하위 4 비트씩 총 56 비트가 키로 사용된다. 이것은 DES의 키 크기 56 비트와

같은 것으로 키의 탐색 영역은 2^{56} = 7.2 × 10^{17}이다. 평문 및 암호문의 길이는 DES가 64 비트이고, HEA는 한글 64 음절 즉, 64×15=960 비트이다. 따라서 한 번의 키 탐색에 소요되는 비용은 HEA가 DES보다 15배이므로, 키 탐색 공격을 더 어렵게 한다.

문자 발생 빈도를 이용하여 키 공간을 줄이는 침입도 있을 수 있다. 한글의 초성과 중성 하위 4 비트의 발생 빈도를 현대 국어의 음소 발생 빈도를 고려하여[12] 구한 결과가 표 3에 나타나 있는데, 이것은 7 비트에 대해 'e'의 발생빈도가 .13으로 가장 높고 'x', 'z'의 발생 빈도가 .00로 가장 낮은, 영문 알파벳과 유사하다고 볼 수 있다[13:p.26]. 즉, HEA의 키로 영문 8 문자 또는 한글 7 음절이 사용될 때, 키 공간을 줄이는 침입에 대한 안전도는, DES 등의 기존 암호 알고리즘의 안전도와 유사하다.

2. Differential Cryptanalysis

DES 단계 i의 키를 K_i, 한 쌍의 오른쪽 입력을 R_i, R_i^*라 할 때, 그 단계 S Box의 input-xor인 $(E(R_i)\oplus K_i)\oplus(E(R_i^*)\oplus K_i)$는 $E(R_i)\oplus E(R_i^*)$가 되어 키가 소거되지만, S Box의 출력을 exclusive-or한 output-xor는 키 값에 의존한다. DES에 대한 differential cryptanalysis(이후 DC라 부르겠음)는 이 사실에 기초하여 chosen plaintext attack으로 키를 찾아내며, 마지막 단계의 input-xor와 output-xor를 알려진 평문과 암호문으로부터 구하기 위하여, 중간 단계의 input-xor, output-xor 중 확률이 높은 것을 이용한다[13-17]. HEA에 대한 DC는 크게 3 가지 방법을 고려할 수 있다. 첫째는, DES에 대한 DC와 같이 한 쌍의 평문을 exclusive-or한 input-xor와 각 평문에 대응하는 암호문들을 exclusive-or한 output-xor를 이용하는 방법이고, 둘째는 한 쌍의 평문을 exclusive-or한 input-xor와 각 평문에 대응하는 두 암호문의 (- mod N) 연산결과인 output-modn을 이용하는 것이며, 셋째는 HEA의 MOD 연산을 XOR로 근사시켜 공격을 수행하는 것이다.

첫번째 방법으로 3 단계 HEA에 DC를 수행해 보자. 단계 3의 S Box에 대한 input-xor는 $E(R_2)$와 $E(R_2^*)$를 exclusive-or한 것으로, 이것은 알려진 값인 $E(L_3)$와 $E(L_3^*)$를 exclusive-or하여 얻을 수 있다. $R_3 = L_2+f(R_2,K_3) \bmod N = R_1+f(R_2,K_3) \bmod N = L_0+f(R_0,K_1)+f(R_2,K_3) \bmod N$이므로, $f(R_2,K_3)$를 A로, R_3-L_0을 B로, $f(R_0,K_1)$를 x로 표시하면, $A = B-x \bmod N$이 얻어진다. 마찬가지로 두 번째 입력에 대하여 $A^* = B^*-x^* \bmod N$이 얻어지며, 이들로부터 output-xor를 $P^{-1}(A\oplus A^*)= P^{-1}((B-x \bmod N) \oplus (B^*-x^* \bmod N))$로 표시할 수 있다. DC를 적용하려면, R_0에 제한 조건을 두어 K_1 항이 있는 x, x^*를 소거시키고 알려진 값인 B, B^*만으로 output-xor가 구성되어야 한다. 그러나, ⊕와 (- mod N)는 교환 법칙이나 결합 법칙이 성립하지 않는 연산들이므로, x와 x^*를 소거시킬 수 없게 된다. 즉, input-xor도 키에 의존하게 되어, DC를 수행할 수 없다.

두번째 고려사항인, 암호문의 쌍에 대하여 output-xor가 아닌 output-modn을 구하는 방법을 알아보자. 이 경우, output-modn인 $P^{-1}((B-x \bmod N) - (B^*-x^* \bmod N) \bmod N)$은, 적당한 제한 조건을 주어 x, x^*를 소거시키고 알려진 값인 B, B^*만으로 구성할 수

초성 하위 4비트값 (대응 초성)	0 ㄱ,ㅌ	1 ㄲ,ㅍ	2 ㄴ,ㅎ	3 ㄷ	4 ㄸ	5 ㄹ	6 ㅁ	7 ㅂ	8 ㅃ	9 ㅅ	10 ㅆ	11 ㅇ	12 ㅈ	13 ㅉ	14 ㅊ	15 ㅋ
발생확률 (총합=1.00)	.13	.03	.12	.10	.00	.03	.07	.06	.00	.14	.00	.16	.09	.00	.04	.02
중성 하위 4비트값 (대응 중성)	0 ㅏ,ㅠ	1 ㅐ,ㅟ	2 ㅑ,ㅡ	3 ㅒ,ㅢ	4 ㅓ,ㅣ	5 ㅔ	6 ㅕ	7 ㅖ	8 ㅗ	9 ㅘ	10 ㅙ	11 ㅚ	12 ㅛ	13 ㅜ	14 ㅝ	15 ㅞ
발생확률 (총합=1.00)	.20	.11	.07	.00	.21	.05	.08	.01	.13	.02	.00	.00	.01	.09	.01	.00

표 3. 현대국어의 음소 빈도 고려한 초성 및 중성의 하위 4 비트 발생 빈도
Fig 3. Frequency rates of lower 4 bits of initial consonant and middle vowel considering the usage of current Korean

는 있다. 그러나, 그림 3에서 보이듯 동일한 Si 규칙을 이용하는 비트들 5 무리가 서로 연관되어 있어서, 확률이 높은 characteristic을 구하기 위하여 구성해야 할 differential distribution table(DDT)[15]의 크기가 input-xor 부분은 $(2^5)^6=2^{30}$, output-modn 부분은 초성의 경우 $19^4\geq(2^4)^4=2^{16}$이다. 따라서 DDT 한 항의 평균 값과 평균 확률이 $2^{30}/19^4$, $(2^{30}/19^4)/2^{30}=1/19^4$로, DES가 갖는 평균 확률 1/16보다 상당히 작다. 또한 mod 연산의 특성상 고른 분포를 보일 것이므로, 확률이 높은 characteristic을 찾기가 어려움을 알 수 있다. 즉, DC가 성공하려면 확률이 높은 characteristic이 필요한데, 이를 찾기가 어려운 HEA는 DC에 대하여 안전하다고 말할 수 있다.

세 번째 방법인 MOD 연산을 XOR로 근사시키는 공격은, HEA의 MOD 연산을 XOR로 치환한 HEAXOR에 대하여 침입을 수행하고 여기에 HEAXOR와 HEA의 차이를 적용하는 순으로 진행된다. 우선, HEAXOR에 대하여는 한 음절의 초성, 중성, 종성이 모두 같은 값을 갖도록 평문을 구성하여 XOR3D 연산에 의한 교란을 없애면, 초성, 중성, 종성 각각에 대하여 32 개의 5 비트 중 i (i=1,...,5)번째 비트들로 이루어진 32 비트가 DES의 수행 결과와 동일해진다. 따라서, DES에 대한 DC와 같은 방법으로, HEAXOR에 대하여 키를 찾을 수 있게 된다.

다음으로, HEA가 각 단계를 수행하며 HEAXOR와 동일한 입출력쌍을 가지려면, MOD 연산 결과와 XOR 연산 결과가 같아야 하고, 또한 XOR3D 연산에 의한 교란이 없어야 한다. 초성과 중성의 경우 MOD 연산이 홀수인 19, 21에 대하여 수행되므로 XOR 연산 결과와 많은 차이를 보이나, 종성의 경우 MOD 연산이 짝수인 28에 대하여 수행되므로, 최하위 비트의 경우 HEA와 HEAXOR의 연산결과는 동일하다. 따라서 종성의 최하위 비트들을 이용하여 키를 찾는 것이 가장 유리한데, 이 경우 XOR3D 연산에 의한 교란이 없는 것은 L 부분이 가질 수 있는 '0'부터 '18'까지 19 가지와, 키 값이 적용된 $f(R,K_i)$ 부분이 가질 수 있는 '0'부터 '31'까지 32 가지의, 총 608 가지 중, 초성의 MOD 연산과 중성의 MOD 연산 결과가 같은 528 가지이다. 즉, 한 비트당 528/608의 확률이므로, HEA와 HEAXOR가 32 음절로 구성된 한 단계에서 동일한 종성의 최하위 비트들을 가질 확률은 $(528/608)^{32}$이다. 한편, HEAXOR에서 종성의 최하위 비트들에 대하여 DC를 수행하는데 필요한 chosen plaintext의 개수는, DES에 대한 DC에 대하여 현재까지 알려진 것과 같은 2^{47}이므로[16], 각 단계가 독립적으로 수행된다고 가정할 때, 두 단계만 고려해도 HEA에 대하여 DC를 수행하는데 필요한 chosen plaintext의 개수는 $2^{47}\times\frac{1}{(528/608)^{32\cdot 2}}>2^{60}$으로 키 전수 탐색의 복잡도보다 높다.

MOD를 XOR로 근사한 후 MOD와 XOR의 차이를 적용하는 침입 방법이, RC5 알고리즘의 경우에는 효과적인데[18,19] 비하여 HEA의 경우에는 그렇지 못한 것은 한 단계에서 덧셈이 한번 발생하는 RC5와 달리 HEA는 독립적으로 32 번 일어나기 때문이다.

3. Linear Cryptanalysis

Linear cryptanalysis(이후 LC라 부르겠음)는 평문과 암호문의 일부 비트들을 exclusive-or한 값이 키의 일부 비트들을 exclusive-or한 값과 같거나 다를 확률이 항상 1/2은 아니라는 사실에 기초하여, 키 전수 탐색보다 더 작은 복잡도로 DES의 키를 찾아낸다[20,21]. 즉, LC의 핵심사항은 linear expression $P[i_1,i_2,\cdots,i_a]\oplus C[j_1,j_2,\cdots,j_b]=K[k_1,k_2,\cdots,k_c]$을 만족하는 확률을 p라 할 때, $|p-1/2|$을 최대로 하는 linear expression을 구성하는 것이다. 여기서, $P[i_1,i_2,\cdots,i_a]$가 의미하는 것은 평문의 $i_1,i_2,\cdots,i_a$번째 비트를 모두 exclusive-or한 값이며, $C[j_1,j_2,\cdots,j_b]$는 암호문의 $j_1,j_2,\cdots,j_b$번째 비트들을, $K[k_1,k_2,\cdots,k_c]$는 키의 $k_1,k_2,\cdots,k_c$번째 비트들을 모두 exclusive-or한 값이다. DES의 대표적인 1 단계 linear

expression은 키를 K, 임의의 입력을 X라 하고, least significant bit의 위치를 0으로 표시할 때, 12/64의 확률로 S Box S_5가 갖는 $X[15]\oplus f(X,K)[7,18,24,29]=K[22]$이 다. 이 때, $X[15]\oplus K[22]$는 주어진 평문 X의 15 번째 비트와 키 K의 22 번째 비트를 exclusive-or한 값으로 S Box S_5의 입력 6 비트 중 하위로부터 4 번째 비트이며, $f(X,K)[7,18,24,29]$는 S Box S_5의 출력 4 비트들을 모두 exclusive-or한 값이다[20].

3 단계 HEA에 위의 expression을 이용하여 LC를 수행하면, 첫번째 단계에서 $P_R[15]\oplus(X2_R-P_L \bmod N)[7,18,24,29]=K_1[22]$를 얻고 세번째 단계에서 $C_R[15]\oplus(C_L-X2_R \bmod N)[7,18,24,29]=K_3[22]$를 얻는다. 여기서, $X2_R$는 두번째 단계 입력의 오른쪽 부분을, P_L,P_R과 C_L,C_R는 각각 평문과 암호문을 말하며, 초성, 중성, 종성별로 각각 식이 구성된다. Linear expression을 구하려면, 이 두 식을 exclusive-or하여 우변은 $K_1[22],K_3[22]$만으로 구성하고, 좌변은 알려지지 않은 항 $X2_R$를 소거하고 알려진 값인 P,C만으로 구성해야 한다. 그러나, DC 부분에서 기술한 것과 마찬가지로 ⊕와 (- mod N)는 교환 법칙이나 결합 법칙이 성립하지 않는 연산들이므로, $X2_R$를 소거할 수 없다. $X2_R$항의 소거를 위하여 linear expression의 개념을 (- mod N) 연산으로 구성해 보려는 시도도 있을 수 있다. 그러나, 이것은 HEA도 DES와 마찬가지로 S Box의 입력을 평문과 키의 exclusive-or값으로 주기 때문에 적용할 수 없다. HEA에 대하여 MOD를 XOR로 근사시키는 침입도 있을 수 있는데, IV-2 절에서 기술한 것처럼 MOD 연산이 단계 별로 32 번 일어나는 HEA의 특성상 별다른 이득을 얻을 수 없다. 즉, HEA는 LC에 대하여 안전하다고 볼 수 있다.

4. 암호문의 임의성 (Randomness of Ciphertext)에 대한 통계학적 검사

안전한 암호 알고리즘은 평문의 구성에 관계없이 임의성이 있는 암호문을 가져야 한다[11,22]. 대부분 동일한 값으로 이루어진 평문(표 4에서 단일 평문)을 암호화한 것과, 임의의 평문을 암호화한 2 가지 종류의 암호문에 대하여 임의성 검사를 수행하였는데, 초성, 중성, 종성 별로 코드 값의 분포범위가 다른 HEA의 특성을 고려하여, 각 검사는 초성, 중성, 종성 별로 수행하였다. 수행한 검사는 빈도 검사(frequency test), 연속 빈도 검사(serial test), 분할 빈도 검사(partition test)로[22,23], 빈도수에 대한 카이제곱 검사를 수행하였다. 표 4에 90 %와 98 % 범주를 벗어나는 빈도들의 백분율 값이 나타나 있는데, 초성, 중성, 종성 별로 임의성을 갖고 있음을 알 수 있다.

		단위:%	초성	중성	종성
빈도 검사	90% 범주위반	단일평문	9.82	9.78	9.83
		임의평문	9.83	9.77	9.78
	98% 범주위반	단일평문	1.96	2.03	2.03
		임의평문	2.03	2.05	1.99
연속 빈도 검사	90% 범주위반	단일평문	9.91	10.11	10.23
		임의평문	9.57	10.41	10.16
	98% 범주위반	단일평문	1.93	1.94	2.03
		임의평문	2.02	2.00	1.93
분할 빈도 검사	90% 범주위반	단일평문	19.36	14.41	8.50
		임의평문	18.59	14.57	8.37
	98% 범주위반	단일평문	4.31	2.97	1.77
		임의평문	3.97	3.17	1.75

표 4. 암호문의 임의성 검사 결과
Table 4. Experimental results of randomness of ciphertext

5. 연쇄효과 (Avalanche Effect) 검사

5.1 평문-암호문 연쇄효과 (Plaintext-Ciphertext Avalanche Effect)

비트 단위 암호 시스템에서의 평문-암호문 연쇄효과(plaintext-ciphertext avalanche effect)는 평문의 한 비트가 변할 때 암호문의 각 비트가 변할 확률이 1/2인 것을 말한다[22]. DES의 S-box는 이 연쇄효과를 만족하도록 설계되었고, HEA는 내부적으로 DES의 S-box를 사용하므로 비트별 연쇄효과는 DES와 유사하다. 추가로, 한글 음소 단위로 동작하는 HEA의 특성을 고려하여 평문의 한 음소가 변경될 때 암호문의 각 음소가 받는 영향을 조사하였다. 이를 위하여 평문 한글 64 음절 P와, P와 한 음소만이

단위 (%)

90 % 범주 위반	10.2
98 % 범주 위반	1.98

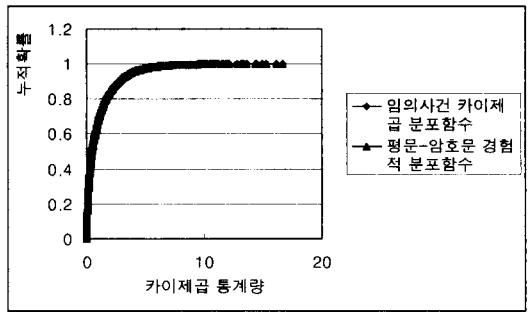
표 5. 평문-암호문 연쇄효과 검사 결과
Table 5. plaintext-ciphertext avalanche effect

다른 P'을 생성하고, 고정된 키에 대하여 이들을 암호화하여 C, C'을 구한다. HEA가 평문-암호문 연쇄효과를 만족한다면, 암호문 C와 C'의 특정 위치의 음소가 같거나 다를 확률은 서로 독립적이어야 한다. 즉, 같은 위치에 있는 초성이 값이 같을 확률은 1/19, 다를 확률은 18/19이며, 중성이 값이 같을 확률은 1/21, 다를 확률은 20/21이며, 종성이 같을 확률은 1/28, 다를 확률은 27/28이다.

이 사실에 기초하여 36864 개의 카이제곱 통계량 V를 구하고 이에 대하여 카이제곱 검사와 콜모고로브-스미노브 검사를 수행하였다[22,23,24]. 콜모고로브-스미노브 (Kolmogorov-Smirnov) 검사는, 임의 사건에 대한 카이제곱 분포 함수 (distribution function= hypothesised cumulative distribution function)와, 발생한 사건의 카이제곱 통계량 V들로부터 구한 경험적 분포 함수 (empirical distribution function = cumulative relative frequency graph)의 차이를 보고, 사건의 임의성을 판단하는 방법이다. 표 5에 주어진 카이제곱 검사 결과는 HEA가 평문-암호문 연쇄효과를 만족함을 보이며, 그림 5에 주어진 임의 사건에 대한 카이제곱 분포함수와 HEA의 평문-암호문 연쇄효과에 대한 경험적 분포 함수의 유사한 형태도 이를 뒷받침한다[22].

5.2 키-암호문 연쇄효과 (Key-Ciphertext Avalanche Effect)

HEA는 내부적으로 DES에 의존하므로 DES가 갖는 비트 단위의 키-암호문 연쇄효과를 갖는다. 추가로, 키는 비트 단위로 의미를 갖으나 암호문은 한글 음소 단위로 의미를 갖는 HEA의 특성을 고려하여, 키의 한 비트가 변경될 때 암호문의 각 음소가 받는 영향을 조사하였다. 이를 위하여 임의의 56 비트 키 K를 생성하고, K와 한 비트만 다른 K'을 구한다. 고정된 평문에 대하여



각 함수는 36864개의 조밀한 점의 분포가 선으로 나타나있다

그림 5. 임의 사건에 대한 카이제곱 분포함수 및 평문-암호문 연쇄효과에 대한 경험적 분포함수
Fig 5. Kai-square distribution function of random event and empirical distribution function of plaintext-ciphertext avalanche effect

키 K에 대응하는 암호문을 C, 키 K'에 대응하는 암호문을 C'이라 하자. HEA가 키-암호문 연쇄효과를 만족한다면, 암호문 C와 C'의 특정 위치의 음소가 같거나 다를 확률은 서로 독립적이어야 한다. 이 사실에 기초하여 10752 개의 카이제곱 통계량 V를 구하고 키-암호문 연쇄효과 검사를 수행하였다. 검사 결과, 전 절에서 기술한 평문-암호문 연쇄효과와 유사하게, HEA가 키-암호문 연쇄효과를 만족함을 알 수 있었다.

V. 성능 분석

수행 연산의 횟수를 고려해 보면, HEA는 한 음절을 구성하는 초성, 중성, 종성 각 5 비트의 같은 비트 위치에 있는 3 가지 값을 비트 별로 exclusive-or하여 이 결과에 EP, XOR, S, P 연산이 수행된다. 즉, XOR3D 연산을 추가로 수행함으로써 EP, XOR, S, P 연산이 처리하는 비트 길이가 DES의 대략 1/3이 되도록 한다.

PEM(Privacy Enhanced Mail)에서는 보낼 평문을 encrypt 과정을 거쳐 암호화하고, 생성된 암호문을 encode 과정을 거쳐 표시 가능한 형태로 바꾼 후 전송을 수행하고, 수신자도 decode를 한 후, 암호문을 해독하게 된다. 이 encode 과정은 3 바이트(=24 비트)의 문자를 6 비트씩 나눈 후 6 비트 당 한 바이트를 할당하여 총 4 바이트로 변환한다

[1,25]. PEM 등과 같이 암호문이 표시 가능해야 하는 응용의 경우, HEA는 암호화 결과가 모두 표시 가능하므로 별도의 변환작업이 필요하지 않으나, DES는 encode 과정에서 한글 64 음절에 해당하는 1024 비트의 변환을 위하여, $\lceil \frac{1024}{6} \rceil = 171$ 번의 변환이 필요하다. 암호문의 평문에 대한 크기 비도 HEA는 입력과 출력의 크기가 동일하므로 1.00이고, DES는 각 6 비트가 1 문자 (8 비트)로 확장되므로 $\frac{8}{6} = 1.33$이다.

VI. 결론 및 향후 연구 계획

본 논문에서는 한글의 다차원 구조를 이용한 한글 암호 알고리즘 HEA를 제안하였다. HEA는 DES에 기반하고 있으며, DES가 갖는 암호문의 음소별 임의성, 평문-암호문 연쇄효과, 키-암호문 연쇄효과를 만족한다. 또한, DES의 취약점으로 알려졌던 키 전수탐색, differential cryptanalysis 및 linear cryptanalysis에 대하여도 어느 정도 안전하다고 말할 수 있다. HEA는 암호화 결과가 모두 표시 가능하므로 암호문이 표시 가능해야 하는 응용에 유용하게 사용될 수 있다. 앞으로, HEA에 대한 좀더 다양한 안전성 분석 및 이에 기초한 알고리즘 개선 작업이 수행되어야 한다.

참고 문헌

[1] Privacy Enhancement for Internet Electronic Mail: Part I-IV, RFC1421-1424, 1993.

[2] 임채호, 류재철, 윤기송, 변옥환, "PEM 서비스를 위한 증명서 기반의 키관리 연구," 통신정보합동학술대회, Vol. 4, pp. 673-677, 춘천, 4월 1994.

[3] Stephen T. Kent, "Internet Privacy Enhanced Mail," *Communications of the ACM*, vol. 36, No. 8, pp. 48-60, August, 1993.

[4] Data Encryption Standard, Fedral Information Processing Standatds (FIPS) Publication 46, National Bureau of Standards, US Department of Commerce, Washington, DC, Jan., 1977.

[5] Rhee ManYoung, *Cryptography and Secure Communications*, pp. 47-102, McGraw-Hill, 1994.

[6] Gilles Brassard, *Modern Cryptology - A tutorial*, Springer-Verlag Lecture Notes in Computer Science **325**, 1988.

[7] 이 진석, "한글 코드의 세 갈림길 - 조합형, 완성형, 유니코드," 건잠머리 유틸리티 개발팀

[8] 변 정용, "훈민정음 원리의 공학화에 기반한 한글 부호계의 발전 방향," 정보과학회지, 제 12권, 제 8호, pp. 72-88, 1994.

[9] Uni Code System 관련 문서, http://www.unicode.org

[10] 한 승조, 이 상호, 구 연설, "완성형 한글을 위한 압축과 암호화 방법," 정보과학회 논문지, 제 20권, 제 9호, pp. 1353-1364, 1993.

[11] Cryptography FAQ (Frequently Asked Questions), http://www.rsa.com/rsalabs/faq, 1998.

[12] 김혜숙, 현대국어의 사회언어학적 연구, IV장 현대국어의 사회적 표현 실태를 통한 한국인의 의식구조, 표9, 표11, (78000 어휘의 첫음절 조사), 태학사, 1991.

[13] Douglas R. Stinson, *Cryptography Theory and Practice*, CRC Press, Inc., 1995.

[14] Bruce Schneier, *Applied Cryptography - Protocols. Algorithms, and Source Code in C*, John Wiley & Sons, Inc., 1994.

[15] Eli Biham, Adi Shamir, "Differential cryptanalysis of DES-like Cryptosystems," *Advances in Cryptology-CRYPTO'90*, Springer-Verlag Lecture Notes in Computer Science **537**, pp. 2-21, 1991.

[16] Eli Biham, Adi Shamir, "Differential cryptanalysis of the Full 16-round DES," *Advances in Cryptology-CRYPTO'92*, Springer-Verlag Lecture Notes in Computer Science **740**, pp. 487-496. 1993.

[17] T.Kaneko, K.Koyama, R.Terada, "Dynamic swapping schemes and differential cryptanalysis," *IEICE Tr. Fundamentals*, v.

E77-A, n. 8, pp. 1328-1335, August, 1994.

[18] A.Biryukov, E.Kusholevitz, "Improved Cryptanalysis of RC5," Eurocrypt98, 1998.

[19] B. Kaliski, Y.Yin, On the security of the RC5 encryption algorithm, RSA lab TR-602, 1998.

[20] Mitsuru Matsui, "Linear cryptanalysis method for DES cipher," *Advances in Cryptology-EUROCRYPT'93*, Springer-Verlag Lecture Notes in Computer Science **765**, pp. 386-397. 1994.

[21] Y.Nakao, T.Kaneko, K.Koyama, R.Terada, "The security of an RDES Cryptosystem against linear cryptanalysis," *IEICE Tr. Fundamentals*, v. E79-A, n. 8, pp. 12-19, January, 1996.

[22] Helen Gustafson, Ed Dawson, Bill Caelli, "Comparison of block ciphers," Advances in Cryptology-AUSCRYPT'90, pp. 208-220, 1990.

[23] D. E. Knuth, *The art of Computer Programming* 2nd Ed., Vol. 2: Seminumerical Algorithms, Addison-Wesley, 1981.

[24] B. Brown, J. Lovato, K. Russel, DCDFLIB (Library of C Routines for Cumulative Distribution Functions, Inverses, and Other Parameters), University of Texas, Feb., 1994.

[25] 서울대학교 컴퓨터신기술공동연구소, 인터넷 보안 시스템 구축 연구 보고서, pp.199-202, 초고속 정보 통신 응용 기술 개발 사업, 정보통신부, 1997.

著者紹介----------

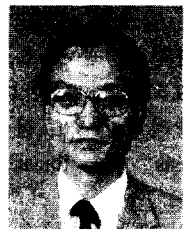
김윤정 (Yoonjeong Kim) 학생회원

1991년 2월 서울대학교 컴퓨터공학과 학사.
1993년 2월 서울대학교 컴퓨터공학과 석사.
1993년 3월 ~ 현재 서울대학교 컴퓨터공학과 박사과정.
<관심분야> 시스템 보안, 암호학, 네트워크 보안.

박근수 (Kunsoo Park) 종신회원

1983년 서울대학교 컴퓨터공학과 학사.
1985년 서울대학교 컴퓨터공학과 석사.
1991년 Columbia University 전산학 박사.
1991년 ~ 1993 년 University of London, King's College 조교수.
1993년 ~ 현재 서울대학교 컴퓨터공학과 조교수.
<관심분야> 알고리즘 설계 및 분석, 병렬 계산, 암호학.

조유근 (Yookun Cho) 정회원



1971년 서울대학교 건축공학과 학사.
1978년 미국 미네소타대학교 전산학과 박사.
1979년 ~ 현재 서울대학교 컴퓨터공학과 교수
1984년 ~ 1985년 미국 미네소타 대학교 교환교수
1993년 ~ 1995년 서울대학교 중앙교육연구전산원장.
1995년 한국정보과학회 부회장.
1999년 서울대학교 공과대학 교무담당 부학장
<관심분야> 운영체제, 알고리즘 설계 및 분석, 시스템 보안.