論 文

온 라인 전송에 있어서 디지탈서명을 위한 압축코딩과 암호코딩의 결합 시스템에 관한 연구

正會員 韓 承 朝* 正會員 李 相 鎬** 正會員 具 然 鳥**

A Study on the Concatenation System of Compression Coding and Secrecy Coding for Digital Signature in On-Line Transmission

Seung-Jo Han*, Sang-Ho Lee**, Yeon-Seol Koo** *Regular Members*

要 約

온 라인 전송에 있어서 정보를 효율적으로 안전하게 전송하기 위해서는 자료압축과 기밀성 그리고 인증성이 요구된다. 이를 위해서 본 논문에서는 LZW를 완성형 한글을 적용하여 압축열이 2개 생성되는 LZWH4를 제안하며, DES를 개선하여 Differential Cryptanalysis에 대한 일부 대응방안으로 SAC과 상관계수에 만족하는 S-box를 S1~S8에서 S1~S16으로 확장하여 HDES1를 설계하며, 온 라인 전송에 있어서 디지탈서명을 위해 이들을 효율적으로 결합한 LZWHDES1를 구현한다. 또한 일반적인 암호강도의 척도인 U.D.(Unicity Distance)에 있어서 HDES1이 DES와 HDES에 비해서 증가됨을 보이며, 본 논문에서 제안한 LZWHDES1가 LZW를 DES에 직접 연결한 LZWDES와 LZWH2를 HDES에 직접 연결한 LZWHDES와 비교하여 효율적으로 수행시간이 단축됨을 보이며, LZWHDES1이 관용키 암호시스템으로써 디지탈서명 시스템에 활용될 수 있음을 보인다.

ABSTRACT

To transmit information efficiently and securely in On-line transmission, data compression, secrecy and authentication are required. In this paper, we propose LZWH4 which creates two compression strings with applying Hnageul to LZW, design HDES1 by extending S-box (S1-S16) which satsfies SAC and correlation coefficient as a partial countermeasure of Differential Cryptanalysis and implement LZWHDES1 which concatenates efficiently these for digital signature in On-line transmission. Also HDES1 is more in U.D.(Unicity Distance) than DES and HDES. We show that the proposed LZWHDES1 reduces processing times than LZWHDES which LZW is directly concatnated to DES and LZWHDES which LZWH2 is directly concatenated to HDES. LZWHDES1 can be used to digital signature system as conventional key cryptosystem.

* 朝鮮大學校 工科大學 電子工學科
Dept. of Elec. Eng., Chosun Univ.
** 忠北大學校 自然科學大學 컴퓨터科學科
論文番號 : 94 2

Ⅰ. 서 론

불과 10년전만 하드라도 전화회선을 이용한 음성통신이 통신의 대부분을 차지했다. 그러나 근래에 와서는 컴퓨터와 통신기술이 하모니를 이루어 컴퓨터 통신이 출현하게 되어, 이제는 과거의 교통수단에 의한 문서수발이 컴퓨터 통신망을 통하여 이루어지게 되었다. 따라서 컴퓨터 통신망을 이용한 통신이 급증하고 일반화됨에 따라 정보의 신속한 교환 및 효율적인 처리가 필요할 뿐만 아니라 정보 전송시 상대방의 신원을 확인하거나 사용자의 정당성을 인증하고 송·수신자간에 일어날 수 있는 재반된 분쟁을 해결할 수 있는 제도나 절차가 필연적으로 이루어져야 한다[1,2]. 그러므로 컴퓨터 통신망을 통하여 정보를 경제적이면서 효율적으로 안전하게 전송하기 위해서는 그림 1.과 같이 압축(효율성)과 암호(기밀성) 그리고 디지탈서명(인증성)이 필요하다. 그러나 압축과 암호 그리고 디지탈서명을 각각 독립적으로 수행한다는 것은 매우 불합리하고 비능률적이므로 이들을 각각 특성에 따라 서로 결합해야 하는데, 이들 결합 중 압축과 암호의 결합에 대한 특성을 이용하여 디지탈서명을 구현하게 하므로써 효율성과 기밀성 그리고 인증성을 동시에 만족하도록 하는 것이다[2,3]. 특히 국내에서는 컴퓨터 통신망에 있어서 한글 테스트 사용량이 증가하고 있으나, 이에 대한 효율성과 안전성을 고려한 디지탈서명을 위해 압축과 암호를 결합한 시스템이 거의 없다. 그러므로 본 논문에서는 한글 텍스트에 대한 압축의 효율성을 최대로 하기 위해서 한글에 대한 문자사용율을 LZW알고리즘[4]에 적용하는 방법으로써 대상자료에 따라 적응적으로 압축하는 LZWH4을 제안하며. 암호에 대한 안전성을 높게하기 위해서 differential cryptanalysis[5]에 공격될 수 있는 DES[6]의 취약점을 보안한 확장된 DES인 HDES1을 설계하여 LZWH4와 효과적으로 결합시켜, 온 라인 전송에 있어서 디지탈서명을 위한 관용키 암호시스템인 LZWHDES1을 구현하고 분석하며, 디지탈서명에 활용될 수 있음을 입증하고자 한다.

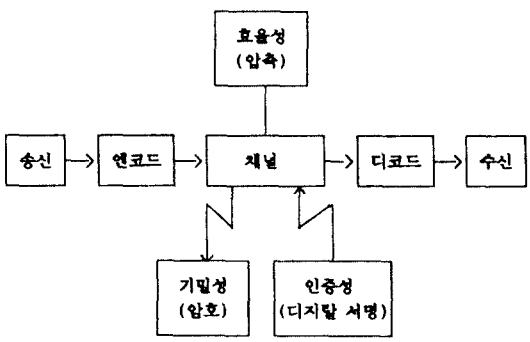


그림 1. 정보 보호 시스템
Fig. 1. The system for information security

Ⅱ. 압축 기법

압축기법을 크게 분류하면 복호 과정에서의 에러를 허용하는 방식인 유잡음 압축(noisy compression 또는 엔트로피 감소(entropy reduction)방식이라 함)과 복호과정에서 에러를 전혀 허용하지 않은 압축방식인 무잡음 압축(noiseless compression 또는 용장성 감소(redundancy reduction)방식이라 함)이 있다[7]. 유잡음 압축방식은 일반적으로 음성, 화상 등과 같이 안정도의 에러율을 허용하는 아날로그 데이터의 압축기법으로 널리 사용하며, 무잡음 압축방식은 에러를 허용하지 않는 텍스트 압축에 사용하는 방식으로 허프만방법, 산술부호화방법 그리고 유니버샬 부호화 방법 등이 있다. 최근에 와서는 여러가지 기법을 합하는 방법으로 LHA가 있는데 이 기법은 LZW 기법으로 1차 압축후 허프만 방법으로 2차 압축하는 방법이다[4,8]. 그러나 이것들은 거의가 외국 압축 프로그램으로 ASCII 코드나 EBCDIC 코드체계 위주로 만들어져 있어서 영문 텍스트나 실행화일에 사용하기에는 압축율이 좋으나 한글 데이터 압축시에는 압축율이 떨어진다. 따라서 사용용도에 따라서 도구를 선택해야 하듯이 한글 데이터 압축에 맞는 기법이 필요하다. 한글은 표음문자로써 초성, 중성, 종성인 음소가 모여 음절을 이루고 있어 다른 언어에서 찾아보기 힘든 특성이 있어서 아직도 많은 논란의 대상이 되고 있지만 완성형 한글인 KSC5601이 표준규격으로 되어 있어서 본 장에서는 완성형 한글 압축에 대해서 논한다. 현재 사용되고 있는 압축기법 중에 한글 텍스트 압축에 가장 용이하게 적용할 수 있는 것이 유니버샬 부호화의 부류인 LZW이다. LZW은 J. Ziv와 A.Lempel이 제안한 후 T.A.Welch[8]가 실용화하여 이름을 따서 명명된 알고리즘으로써 입력 데이터에 대한 정확한 모델링이 요구되는 2패스 부호화 방법인 통계적 기법(허프만방법, 산술부호화방법)과는 다르게 입력 데이터에 대한 사전 통계가 필요하지

않으며 압축 및 복호 이외의 정보가 전송되지 않아 효율적이며, 스트링 포인터만을 아웃코딩하여 전송하는 1패스 부호화방법으로 압축율이 우수할 뿐만 아니라 수행시간이 2패스 부호화방법보다 짧아 실시간 통신환경에 사용되며, 이미 UNIX 유틸리티 및 CCITT V.42bis로댐에서 사용되고 있기 때문에 LZW을 한글 텍스트 압축기법에 적용했다. LZW을 완성형 한글에 적용하기 위해서는 초기 스트링 테이블(0~255)에 압축된 ASCII코드 이외에 2350자를 추가로 등록시킴으로써 입력 스트링 테이블 포인터가 2606부터 시작됨으로 스트링을 나타내고 있는 포인터의 비트 길이가 증가하고 출력 파일의 길이가 증가해 오히려 압축율이 떨어진다. 따라서 이러한 문제점을 보완하는 방법으로 논자는 LZWH2[19]를 발표했다. 이 LZWH2는 한글 사용빈도에 대한 문자 사용을 조사하여 한글을 사용빈도분포에 따라 확률적으로 처리하는 방법으로 사용빈도가 많은 한글은 2바이트로 인식하며 사용빈도가 적은 한글은 영문과 마찬가지로 ASCII코드로 인식하므로써 한글을 단순히 1바이트로 처리하는 방법과 무조건 2바이트로 처리하는 방법의 단점을 보완하는 방법으로 완성형 한글에 있어서는 LZW보다 압축율이 우수함을 보였다. 그러나 LZWH2는 통계적인 기법을 내포하고 있으므로 만일 입력 데이터가 사용빈도분포가 적은 한글로만 구성되어 있다면 압축율이 떨어질 뿐만 아니라 비효율적으로 수행시간이 길어질 것이다. 또한 압축과 암호를 결합하는 방법으로는 LZWH2로 압축한 후 확정된 DES인 HDES로 암호화한 결합 시스템인 LZWHDES는 압축될 전체를 암호화하기 때문에 압축으로 인한 수행시간이 지연되어 온 라인 전송에 있어서는 압축에 의한 효율성을 전혀 기대할 수 없다. 그러므로 효율적이면서 암호화 알고리즘에 효과적으로 결합할 수 있는 LZWH4를 설계한다.

2.1 LZWH4 설계

LZWH2를 개선하여 LZWH4를 설계하는 이유는 다음과 같다.

(1) LZWH2는 통계적 성질을 이용하므로 입력 데이터에 대한 한글 빈도분포에 따라 압축율의 변화가 심하다.

(2) LZWH2를 암호화 알고리즘에 결합하였을 경우에는 수행시간의 지연으로 인하여 실시간 처리에 사용할 수 없다.

(3) LZWH2가 암호화 알고리즘과 결합하여 송신자에 대한 디지탈서명을 할 수 없다. 따라서 LZWH2를 암호화 알고리즘에 결합하였을 경우에는 기밀성만 유지될 뿐 효율성과 인증성을 해결할 수 없다.

그러므로 위와 같은 조건을 만족하도록 LZWH4를 설계한다. 그림 2.는 LZW을 완성형 한글에 적용하는 LZWH4에 대한 스트링 테이블 구성이다.

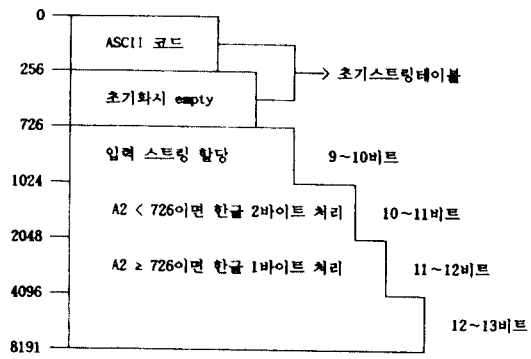


그림 2. LZW 4에 대한 스트링 테이블 공간 구성
Fig. 2. The memory composition of the string table for LZWH4

LZWH4는 통계적 기법과 적응적 사전기법을 혼합시켜 압축코딩을 암호코딩과 효과적으로 결합하기 위한 방법으로 한글에 대한 문자 사용율에서 사용빈도수가 가장 많은 470자를 압축이 수행되기 전에 초기 스트링테이블 영역인 256~725에 등록하지 않고 압축시에 입력되는 한글종류를 초기 스트링 테이블에 256부터 등록시키면서 동시에 입력스트링테이블을 구축해 나가는 방법으로 압축과정은 다음과 같다.

(1) 스트링테이블은 초기스트링테이블(0~725)과 입력 스트링테이블(726~8191)로 나누어지며 다시 초기스트링은 0~255와 256~725영역으로 할당된다. 0~255는 압축전에 ASCII코드값이 미리 등록되어 있으나 256~725는 초기화시에 등록하지 않고 비워두며 입력 스트링테이블의 카운터 A1을 726으로 초기화 시킨다.

(2) 입력 스트링테이블은 prefix (2바이트)와 suffix (2바이트)로 구성되어지며 prefix는 스트링테이블의 포인터만을 저장하며 suffix는 단일 스트링의 코드만을 저장한다.

(3) 입력 데이터에 대한 한글 종류를 순서대로 등록하기 위해 초기스트링테이블에 카운터 A2를 256으

로 초기화 시킨다.

(4) 입력 데이터에서 2바이트를 읽어드려 왼쪽바이트가 C8이하 이면서 MSB가 1이면 한글 코드로 인식되므로 문자 사용빈도수에 관계없이 A2을 검색한다. A2가 725보다 적으면 입력된 한글을 등록시키고 A2를 1씩 증가시키면서 1)를 수행하며, A2가 725일 경우에는 초기스트링테이블에 입력된 한글을 등록할 수 없으므로 2)를 수행한다. 만약에 왼쪽바이트가 C9 이상이거나 혹은 MSB가 0이면 영문이나 한자 또는 특수문자로 인식하여 3)을 수행한다.

1) 등록된 스트링포인터만을 A1의 prefix에 저장한다.

2) 입력된 한글이 256~725영역에 등록되어 있지 않으므로 한글을 상위바이트와 하위바이트로 나누어 영문처럼 처리한다.

3) 왼쪽바이트는 영문 혹은 특수문자이므로 ASCII 코드값을 prefix에 저장한 다음 A1을 1씩 증가시키고 왼쪽바이트를 검색한다.

(4) suffix에 단일 문자코드를 저장하고 (3)의 과정을 반복한다.

(5) (3)과 (4)의 과정을 반복하여 A1이 8191이 되면 726부터 재구축을 시작한다. 그러나 재구축하므로써 이전에 구축된 내용이 지워지더라도 (3)과 (4)의 과정과 동시에 아웃코딩이 실행되어 압축열이 생성되므로 순시복호가능부호화가 된다. 여기서 A1을 8191로 제한된 이유는 과정길이가 길면 검색에 대한 파싱길이가 길어지고 짧으면 압축율이 떨어지는 상관관계에 있으므로 A1을 8191로 제한한다. 원문이 다음과 같이 "예제"일 때 스트링테이블의 공간구성은 표 1.과 같다.

예제 "정보정보사봄보정정보사회ababc"

표 1. LZWH4의 스트링테이블 공간구성

Table. 1. The memory composition of string table for LZWH4.

(a) 초기 스트링테이블

(a) The initial string table

97	[a]
98	[b]
99	[c]

256	[정]
257	[보]
258	[사]
259	[봄]
260	[회]

(b) 입력 스트링테이블

(b) The input string table.

	Prefix (2바이트)	Suffix (2바이트)
726	(정) →256	[보]
727	(보) →257	[정]
728	(정보) →726	[사]
729	(사) →258	[봄]
730	(봄) →259	[보]
731	(보정) →727	[정]
732	(정보사) →728	[회]
733	(회) →260	[a]
734	([a]) → 97	[b]
735	([b]) → 98	[a]
736	([a][b]) →734	[c]
737	([c]) → 99	[ ]

표 1.에서 "[ ]"은 코드값을 의미하며 "( )"은 포인터를 의미하며 "([ ])"는 코드값이 포인터 값임을 의미함. 입력 스트링테이블에서 압축열을 생성하기 위한 아웃코딩은 가변코드길이 개념[10,11]을 이용하여 prefix내의 포인터 값에 따라 [log A1]비트 또는 [log (A1+1)]비트로 압축열을 생성해 간다. 그러므로 A1이 726~1023이면 9비트 또는 10비트로 코딩하며, 1024~2047이면 10비트 또는 11비트로 코딩하며, 2048~4095이면 11비트 또는 12비트로 코딩하며, 4096~8191이면 12비트 또는 13비트로 코딩한다. ([C]는 C 이상의 최소 정수값을 의미함)

LZWH4는 2개의 압축열이 생성된다. 하나는 LZWH2와 같이 prefix를 아웃 코딩하여 압축열(CP : Compression for Prefix)이 생성되며 다른 하나는 256부터 A2에 저장된 한글문자종류를 다음의 "예제"와 같이 12비트 단위로 코드화되어 압축열(CIST : Compression for Initial String Table)이 생성된다.

예 제 : 정(C1A4)는 12비트인 $(011010001000)_2$으로 2진 코드화 된다.

$$C1 : 13 = C1\ (\text{Mod } B0)_{hex} \longrightarrow (01101)_2$$
$$A4 : 04 = A4\ (\text{Mod } A0)_{hex} \longrightarrow (0001000)_2$$
$$C1A4 \longrightarrow (011010001000)_2$$

표 2.은 압축효율을 높이기 위해서 가변코드길이 개념을 사용한 아웃코딩에 의한 압축열 CP와 CIST를 바이트 단위로 나타낸 것이다.

표 2. 아웃코딩에 의한 압축열
Table. 2. Compression string by outcoding

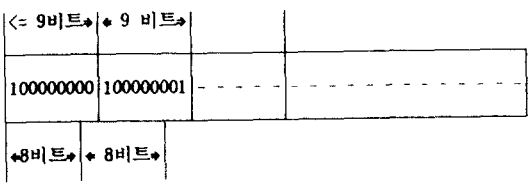
(a) LZWH2의 prefix에 대한 가변코드 길이
(a) Variable code length for prefix of LZWH2

AI	Prefix	Prefix의 2진 코드화	가변코드길이(비트)
726	606	1100001000 (904)	10
727	478	1100000111 (775)	10
728	726	1111111110 (1022)	10
729	506	1100100001 (801)	10
730	179	010110011 (179)	9
731	232	011101000 (232)	9
732	727	1111111011 (1019)	10
733	728	1111111011 (1019)	10
734	717	1111101111 (1007)	10
735	97	001100001 (97)	9
736	98	001100010 (98)	9
737	735	1111111110 (1022)	10
738	99	001100011 (99)	9

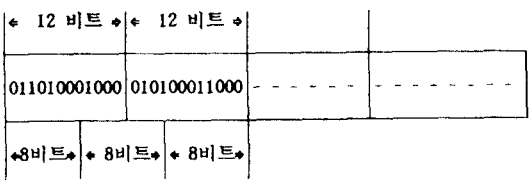
(b) LZWH4의 prefix에 대한 가변코드 길이
(b) Variable code length for prefix of LZWH4

AI	Prefix	Prefix의 2진 코드화	가변코드길이(비트)
726	256	100000000 (256)	9
727	257	100000001 (257)	9
728	726	1111111110 (1022)	10
729	258	100000010 (258)	9
730	259	100000011 (259)	9
731	727	1111111100 (1020)	10
732	728	1111111100 (1020)	10
733	260	100000100 (260)	9
734	97	001100001 (97)	9
735	98	001100010 (98)	9
736	734	1111111110 (1022)	10
737	99	001100011 (99)	9

(c) 아웃코딩에 의한 압축열 CP
(c) Compression string (CP) by outcoding



(d) 아웃코딩에 의한 압축열 CIST
(d) Compression string (CIST) by outcoding



(e) 압축열(CP, CIST) (바이트 단위)
(e) Compression string (CP, CIST)

CP	80	40	7F	E8	14	0F	FC	FF	20	86	13	17	FC	63
CIST	68	85	18	5C	71	DD	C1	80						

그러므로 LZWH4가 LZWH2와 다른 것은 원본에 따라 서로 다른 CIST가 생성 되는데 CIST가 없으면 복호가 전혀 불가능하다는 것이다. LZWH4은 CIST로 인하여 최대 705바이트까지의 오버헤드를 가질 수 있다. 그러나 원본에 사용된 한글 종류가 470자 이내에서는 한글을 2바이트만으로 처리하기 때문에 사용빈도수에 따라 2바이트 혹은 1바이트로 처리하는 LZWH2의 압축열 보다 LZWH4의 CP가 줄어든다. 따라서 동일한 "예제"를 가지고 LZWH2와 LZWH4를 비교하여 보면 표 2.의 (a)와 (b)에서 보인 바와 같이 LZWH2는 738개의 스트링테이블이 구축되나 LZWH4는 737개가 구축되며, LZWH2의 prefix에 대한 전체 코드길이는 135비트인 반면에 LZWH4는 112비트로 구성되어 LZWH2보다 LZWH4의 CP가 줄어듦을 보인다. 그러나 LZWH4는 CIST라는 압축열이 생성되기 때문에 오버헤드가 부가되지만 CIST는 데이타인증을 위한 송수신자 서명으로 사용할 수 있는 장점이 있다. 복호시는 압축과정에 대한 역 과정을 취하므로써 완전히 복호가 되는 것으로 가변코드길이 개념을 이용하여 압축열을 처리하면 포인터가 구해지며 포인터를 prefix에 구축하면 중분분해하여 마지막 포인터에 있는 낱알문자 코드값을 suffix에 저장하므로써 압축시에 사용되었던 동일한 입력스트링테이블이 구축된다.

III. 암호 알고리즘

암호 알고리즘은 크게 관용키 암호시스템과 공개키 암호시스템으로 분류되는데 관용키 암호시스템은 수행속도가 빠르기 때문에 실시간 통신환경에서 사용하기에 적합하지만 인증성을 해결할 수 없다는 단점이 있는 반면에, 공개키 암호시스템은 기밀성 및 인증성은 해결할 수 있으나 수행속도가 아주 느려 온라인 전송에 사용한다는 것이 불가능하다[11]. 그러므로 실시간 통신환경에서는 관용키 암호시스템을 사용하는데, 그 중에서도 세계의 여러나라에서 국가의 표준 암호알고리즘으로 DES를 가장 많이 사용하고 있다. 따라서 본 장에서는 온라인 전송에 있어서 DES

의 암호 강도를 높이기 위해 DES를 확장하려고 한다.

3.1 DES에 대한 암호강도 및 암호해독

오늘날 실질적인 암호강도의 개념은 암호해독을 하는데 얼마만큼의 시간과 경비가 소요되는가를 나타내는 시간 복잡도와 공간 복잡도에 기반을 두고 있으며, 일반적인 암호강도에 있어서는 암호해독을 하는데 필요한 암호문의 최소길이인 U.D.를 암호강도의 척도로 나타내기로 한다. 암호해독방법에는 암호알고리즘을 알고 있더라도 제 3자가 만들어낸 임의의 암호문만을 분석하여 키를 찾아내는 ciphertext-only attack과 특정한 암호문과 이에 대응되는 평문을 이미 알고 가능한 키를 찾아내는 known plaintext attack이 있으며, 임의로 선택할 수 있는 암호문과 이에 대응되는 평문을 알고 비밀키를 찾아내는 chosen plaintex attack이 있다. 공격 방법에는 known plaintext attack하에서 모든 가능한 키로 암호화 하거나 복호화하므로써 키를 찾는 exhaustive공격 방법[11]이 있는데 이 방법의 시간 복잡도는 O(n)이며 공간 복잡도는 O(1)로서 DES에 있어서는 $7*10^{10}$ sec가 소요되며, chosen-plaintext attack하에서 선택한 평문에 가능한 키의 전부를 적용해 사전계산을 하여 생성된 암호문을 키와 함께 메모리 테이블을 구성하는 테이블 lookup공격방법이 있는데, 이 방법의 시간 복잡도는 O(1)이며 공간 복잡도는 O(n)으로서 DES에 대한 공격에 있어서는 $4*10^8$비트의 공간이 소요된다. 또한 테이블 lookup공격방법의 단점을 보안한 방법으로 time-momory trade off공격 방법[15]이 있는데, 시간 복잡도와 공간 복잡도가 각각 $O(n^{2/3})$으로 메모리 공간이나 검색시간을 $O(n^{1/3})$만큼 줄일 수 있다. DES에는 암호함수내에 그림 3.과 같이 S-box가 있는데 S-box설계 기준으로 SAC(Strict Avalanche Criterion)와 상관계수[16]가 있다. SAC[16]는 Wester과 Travares가 제안한 것으로 암호 함수의 각 출력 비트들은 입력 비트들의 모든 변화에 바꾸어질 수 있는 확률이 1/2이어야 한다는 의미로 다음과 같은 조건식을 나타낼 수 있다.

$$\sum_{x\in Z_2^n} f(x)\oplus f(x_i) = (2^{n-1}, 2^{n-1}, 2^{n-1}, \cdots\cdots, 2^{n-1}) = (y_1, y_2, \cdots\cdots, y_i) \qquad (3.1)$$

여기서 $x_i = x \oplus c_i$이며, c_i는 i번째 비트만 1이고 나머지 비트는 모두 0인 n비트 베터이다. 그러므로 i번째

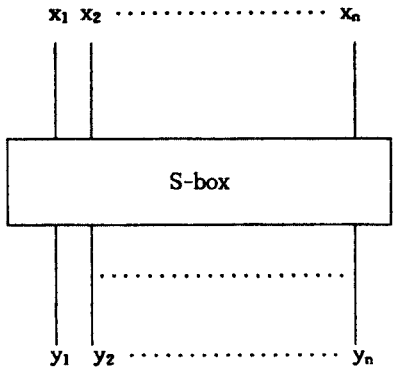


그림 3. S-box
Fig. 3. S-box.

입력 비트가 complement 될 때 j번째 출력 비트가 바꾸어질 확률이 $P_{i,j} = x_i/2^n$라 하면 S-box는 $P_{i,j}$가 0.5에 접근할수록 SAC조건에 강한 S-box가 된다. 또한 S-box는 SAC조건을 만족하더라도 입력 비트와 출력 비트간에 상관관계[15]를 갖지 않도록 상호 독립적이어야 한다. S-box의 출력 비트간의 상관관계는 어느 입력의 한 비트가 complement 됐을 때 출력의 각 비트는 서로간에 상관관계를 갖고 변화하는가를 측정한 것으로 상관계수($-1\le\rho\le1$)가 있는데 상관계수는 0에 접근할수록 잘 설계된 것이다.

3.2 Differential Cryptanalysis

앞 절에서 기술한 바와 같이 암호알고리즘에 따라 다양한 공격방법이 있지만 1992년에 Biham과 Shamir에 의해 DES를 효율적으로 공격하는 방법이 발표[17]되었는데, 이 새로운 형태의 공격방법이 differential cryptanalysis이다. 이 공격방법은 chosen-plaintext공격하에서 적용되는 것으로 암호함수에 대한 평문쌍들의 차이(입력XOR)로부터 암호문 쌍들의 차이(출력XOR)를 통계적인 방법을 통하여 분석하므로써 비밀키를 찾는 방법이다. 입력 XOR이란 2개의 S-box간에 입력되는 쌍들을 EX-OR(Exclusive-OR)한 값이고, 출력 XOR이란 출력되는 쌍을 EX-OR한 값을 의미하며, 각 S-box에 대한 입력 XOR와 출력 XOR간에 있어서 XOR 분포테이블[5,17]이 구성되는데, XOR의 성질 때문에 각 엔트리는 일정한 분포를 갖을 수 없으며 또한 키와 관계없이 임의의 고정된 분포의 값을 갖는다. 이러한 성질을 이용하여 XOR 분포테이블내의 확률이 높은 입력 XOR와 출력 XOR를 알고 있다면 암호함수에 대한 키 값

은 높은 확률로 예측할 수 있으므로 비밀키를 찾을 수 있다. 즉 S1-box에서 키와 입력되기 전의 입력쌍이 1과 35이고 입력 XOR이 34일 때 출력 XOR이 D인 가능한 키는 S1-box의 XOR분포테이블을 이용하여 표 3.과 같이 8/64의 확률로 예측할 수 있다.

표 3. 입력쌍이 1과 35이고 출력 XOR이 D일 때의 가능한 키
Table. 3. Possible keys when the output XOR is D with the input pair (1, 35).

S1-BOX 입력	가능한 키
06, 32	07, 32
10, 24	11, 25
16, 22	17, 23
1C, 28	1D, 29

이러한 방법[15,17]을 이용한 DES의 공격은 최종라운드부터 시작한다. 최종라운드의 암호함수의 입력 XOR은 암호문의 오른쪽 블럭인 32비트와 동일하므로 최종라운드의 암호함수에 대한 출력 XOR를 알면 XOR 분포테이블의 높은 확률을 이용하여 공격이 가능하다. 따라서 최종라운드의 암호함수에 대한 출력 XOR를 N라운드의 특성을 이용하여 구하면 최종라운드의 암호함수에 적용된 키를 높은 확률로 찾아낼 수 있다. 그림 4.은 DES에 있어서 XOR 분포테이블을 이용한 N라운드 특성 중 3라운드 특성유형[17]이다.

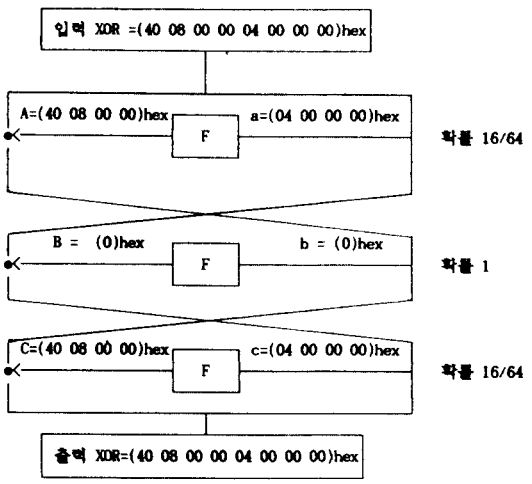


그림 4. XOR 분포테이블을 이용한 3라운드 특성
Fig. 4. The 3 round characteristic using the XOR distribution table.

3.3 HDESI 설계

암호 알고리즘 중에서 가장 일반적으로 사용되고 있는 DES[9]는 1977년 미국 NBS(National Bureau of Standard)에서 공표된 이래 DES에 대한 분석과 분석법에 의해 끊임없이 논란이 되어 왔다. 그러나 최근에 와서는 각 나라에서는 자국 나름대로 독자적인 암호알고리즘을 개발하려고 노력하고 있으며, 일본과 호주에서는 DES와 유사한 FEAL과 LOK1를 개발하여 자국내에서 활용하고 있고, 통합을 눈 앞에 둔 유럽에서도 DES와 유사한 알고리즘이나 새로운 알고리즘을 개발하려고 노력하고 있다. 특히 ISO/IEC TC1/SC21에서는 안정성 서비스를 해결하기위해 ISO 7498-2를 발표하였다. 따라서 우리나라에서도 무조건 통신량만 증가시킬 것이 아니라 이제는 정보 보호 차원에서 새로운 알고리즘을 개발해야 한다. 이에 발 맞추어 DES를 확장하여 일반적으로 암호강도가 증가되는 HDES[18]를 발표했다. 본 절에서는 HDES를 보완하여 HDES1를 제안하고자 한다. DES는 반도체기술과 컴퓨터기술의 발달로 인하여 IC의 패킹밀도와 수행속도가 증가하여 시간복잡도와 공간복잡도에 대한 암호강도가 떨어지고 있으며, exhaustive공격과 테이블lookup공격 그리고 time-memory trade off공격에 대해서 56비트 키길이를 갖는 DES는 암호강도가 감소되며, 2^{37}의 복잡도로 공격이 가능한 differential cryptanalysis공격에 대응하기 위해서는 수행시간이 증가되지 않는 범위에서 암호 알고리즘체계를 변경해야하며, SAC와 상관계수 조건을 만족하며 XOR분포가 균일한 S-box를 재설계해야 하며, S-box수를 증가시켜야 한다. DES는 대치(permutation)와 치환(substitution)이 반복되는 알고리즘으로 다음 식과 같은 EX-OR의 논리적 특성 때문에 만약 키 Ks와 키 Kp가 같다면 암호화 과정과 복호화 과정이 대칭적인 알고리즘이다.

$$(P \oplus Ks) \oplus Kp = P \qquad (3.2)$$

P : 평문

DES는 원문의 한 블럭을 64비트씩 읽어 들여 그림 5.에서와 같이 이를 2개의 32비트 서브블럭(L, R)으로 나누어 각 라운드마다 암호함수를 적용한 것으로 암호화 과정은 다음과 같다.

암호화

$L_n = R_{n-1}$

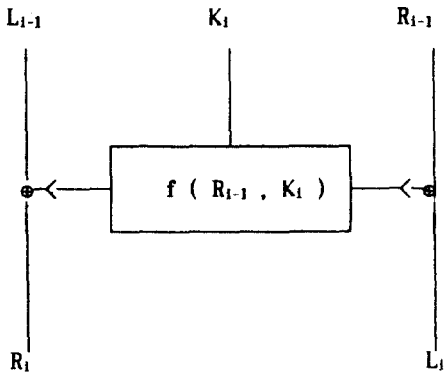


그림 5. 암호화 과정
Fig. 5. Cryptographic procedure

$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i) \tag{3.3}$$

그러므로 식(3.3)에 대해서 EX-OR의 논리적 특성을 이용하면 복호화 과정이 된다.

$$\begin{aligned} R_{i-1} &= L_i \\ L_{i-1} &= R_i \oplus f(R_{i-1}, K_i) \\ &= R_i \oplus f(L_i, K_i) \end{aligned} \tag{3.4}$$

따라서 위와 같은 동일한 특성을 가지면서 원문의 한 블럭을 96비트로 읽어드려 이를 3개의 서브블럭(A, B, C)로 나누어 각각 양쪽에서 암호함수를 적용하여, 각 서브블럭이 16라운드를 반복하도록 하는 암·복호쌍에 대한 가능한 결합수는 9개이나 이들 중에 유일하게 1개 만이 대칭적이며, 마지막 라운드에서 두 서브블럭을 서로 교환하여 암호화 과정과 복호화 과정이 동일한 알고리즘을 갖는 관계식은 다음과 같다.
암호화

$$\begin{aligned} A_i &= C_{i-1} \oplus f(B_{i-1}, K_i) \\ B_i &= A_{i-1} \oplus f(B_{i-1}, K_i) \\ C_i &= B_{i-1} \end{aligned} \tag{3.5}$$

그러므로 식(3.5)에 대한 식(3.2)의 논리적 특성을 이용하면 복호화 알고리즘 공식이 유도된다.
복호화

$$\begin{aligned} A_{i-1} &= B_i \oplus f(C_i, K_i) \\ B_{i-1} &= C_i \\ C_{i-1} &= A_i \oplus f(C_i, K_i) \end{aligned} \tag{3.6}$$

그러나 키를 56비트에서 112비트로 증가시키기 위해서 입력된 키는 128비트에서 패리트 비트인 16비트를 제외한 112비트가 각각 56비트씩 K1과 K2로 양분되고 키 스케줄을 통하여 서브키인 $K_{1,i}$와 $K_{2,i}$가 생성된다. 따라서 112비트 키를 적용하면 다음과 같은 암호화와 복호화 공식이 유도된다.
암호화

$$\begin{aligned} A_i &= C_{i-1} \oplus f(B_{i-1}, K_{1,i}) \\ B_i &= A_{i-1} \oplus f(B_{i-1}, K_{2,i}) \\ C_i &= B_{i-1} \end{aligned} \tag{3.7}$$

복호화

$$\begin{aligned} A_{i-1} &= B_i \oplus f(C_i, K_{2,i}) \\ B_{i-1} &= C_i \\ C_{i-1} &= A_i \oplus f(C_i, K_{1,i}) \end{aligned} \tag{3.8}$$

S-box는 암호함수 f 내에 존재하며 암호함수는 식(3.8)과 같이 $K_{1,i}$와 $K_{2,i}$에 따라 2개로 양분되기 때문에 S-box를 S_1~S_8에서 S_1~S_{16}으로 증가시키기 위해서는 S_1~S_8과 S_9~S_{16}으로 양분하여 2개의 암호함수 f에 다음 식과 같이 적용되도록 한다.

$$\text{암호화} \quad \begin{cases} f(B_{i-1}, K_{2,i}) = P(S_1(b_1), \cdots\cdots, S_8(b_8)) \\ f(B_{i-1}, K_{1,i}) = P(S_9(b_9), \cdots\cdots, S_{16}(b_{16})) \end{cases} \tag{3.9}$$

$$\text{복호화} \quad \begin{cases} f(C_i, K_{2,i}) = P(S_1(b_1), \cdots\cdots, S_8(b_8)) \\ f(C_i, K_{1,i}) = P(S_9(b_9), \cdots\cdots, S_{16}(b_{16})) \end{cases} \tag{3.10}$$

여기서

$$\begin{aligned} b_1, b_2, b_3, \cdots\cdots, b_8 &= E(B_{i-1}) \oplus K_{2,i} \\ b_9, b_{10}, b_{11}, \cdots\cdots, b_{16} &= E(B_{i-1}) \oplus K_{1,i} \end{aligned} \tag{3.11}$$

이며,

$f(B_{i-1}, K_{1,i})$: 암호 함수
$P(S_1 \cdots\cdots)$: P-box

S(b) : S-box

b_i : 6비트 블럭으로 S-box의 입력이 됨

$E(B_{i-1})$: 확장 permutation

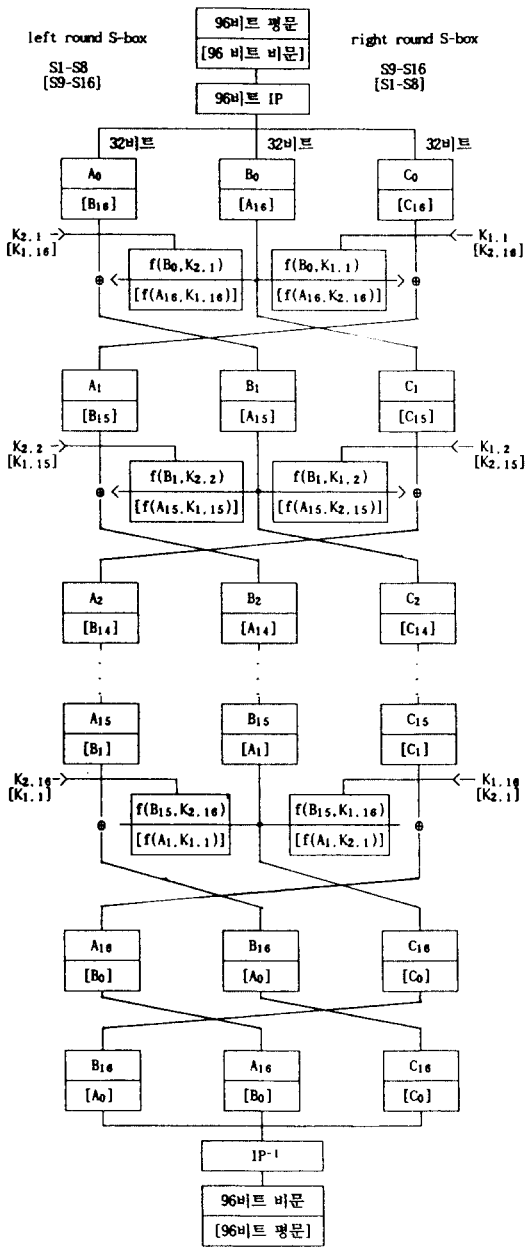


그림 6. HDES1 알고리즘

Fig. 6. The HDES1 algorithm

이다.

따라서 상기에서 분석한 모든 식들을 적용하여 도해하면 그림 6.과 같이 HDES1알고리즘을 표현할 수 있으며, 이 것은 복호화 알고리즘을 의미한다. 복호시는 다만 키를 역순인 $K_{1,16}$, $K_{1,15}$, $K_{1,14}$, ... $K_{1,1}$ 순으로 입력하되 좌우 $K_{1,i}$와 $K_{2,i}$를 교환하여 주어야 한다. DES와 HDES1에 있어서 differential cryptanalysis 공격에 대한 일부 대응 방안으로 N라운드 특성이 구성된 확률을 낮추기 위해 각 서브 블럭에서 16라운드 동안 수행되는 암호함수의 반복횟수를 비교하면 그림 7.과 같다. 그림 7.에서 보는 바와 같이 DES는 서브 블럭(R, L)에서 수행되는 암호함수의 반복횟수는 각각 8회이나 HDES1는 A0에서 B16까지 수행되는 동안 11회 반복하며, B0에서 C16까지 수행되는 동안 10회 반복하며, C0에서 A16까지 수행되는 동안 11회 반복하므로 각 서브블럭(A, B, C)에 따라 암호함수의 반복횟수가 각각 다르다. 또한 HDES는 S-BOX가 S8에서 S16으로 증가하였으므로 N라운드 특성이 구성된 확률을 낮추기 위해서는 DES보다 더 향상 된다고 볼 수 있다. H/W 구현시에는 HDES는 3개의 서브 블럭(96비트)을 동시에 실행하므로 2개의 서브블럭

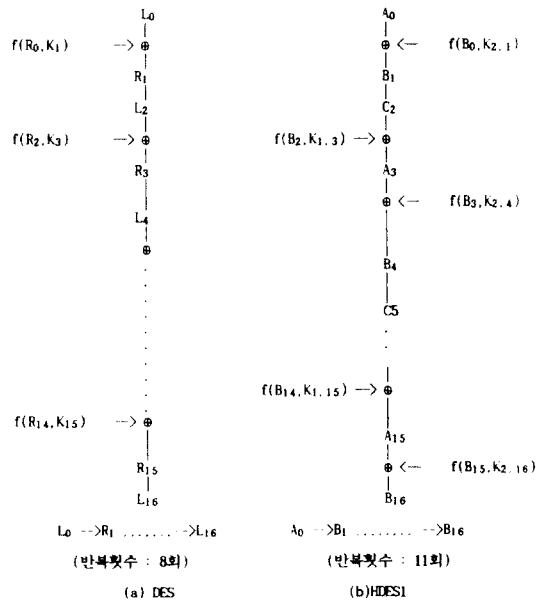


그림 7. DES와 HDES1에 대한 암호함수 절차

Fig. 7. The procedure of secure function for DES and HDES1

(64비트)를 실행하는 DES보다도 오히려 훨씬 수행 속도가 짧아져서 실시간 통신 환경에 매우 적합하며 복호시 알고리즘이 암호시 알고리즘과 동일하기 때문에 암호화 알고리즘과 복호화 알고리즘을 각각 별도로 설계할 필요가 없다.

Ⅳ. 디지탈서명을 위한 압축과 암호 결합

압축과 암호의 결합시스템을 이용하여 디지탈서명을 구현할 수 있다는 것은 컴퓨터 통신에서 효율성과 기밀성 그리고 인증성을 동시에 만족시킬 수 있다는 것이다. 이러한 효과를 성취하기 위해 압축과 암호를 결합하려는 성질에는 다음과 같은 특징이 있다.

(1) 압축이란 용장성을 감소시키는 방법의 일종으로 압축기법에 따라 원문에 포함된 통계적 성질을 압축열에서는 적게 가지고 있으므로 압축열 자체가 암호성질을 지니게 되므로 압축알고리즘이 공개되지 않을 경우에는 비인가자는 압축열을 해독할 수가 없다.

(2) 2패스 방법인 허프만 알고리즘이나 산술 부호화 방법 등은 압축열이 원문에 대한 통계적 성질을 가지고 있어 암호공격에 쉽게 해독될 수 있으므로 압축알고리즘 자체가 암호 성질을 포함하고 있지 않으나, 1패스방법이며 적응적 사전 방식인 LZW은 압축열이 스트링 테이블의 포인터만으로 구성되어 있으므로, 압축열에 원문의 통계적 성질과 용장성이 거의 포함되어 있지 않아 압축알고리즘 자체가 강한 암호성질을 부여하고 있다.

(3) 2장에서 설계한 LZWH4는 2개의 압축열(CP, CIST)를 생성하는데, CIST가 없으면 원문을 생성할 수 없으므로 CIST를 암호화하여 디지탈서명의 해쉬 함수로 사용할 수 있다. 이러한 특징들을 이용하여 온 라인 전송에 있어서 디지탈서명을 위해 압축과 암호를 결합한 방법이 LZWHDES1이며, 구성은 그림 8.과 같다.

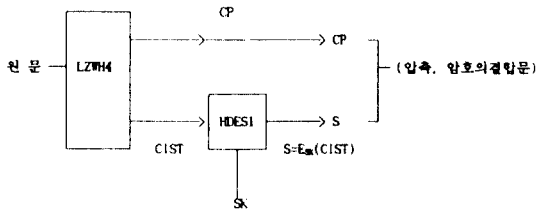


그림 8. LZWHDES1의 구성
Fig. 8. The composition of LZWHDES1.

LZWH4의 압축열인 CP는 스트링테이블의 포인터로 구성되어 있어서 CIST를 알지 못하면 원문의 통계적 성질을 전혀 알지 못한다. 한글에 대한 초기 스트링테이블 영역이 256~725까지이므로 LZWH4가 공개된 압축알고리즘이라도 비인가자가 CIST를 알지 못하고 CP를 해독할 수 있는 복잡도는 $2,350\ P_r$ (여기서 r는 원문의 한글 종류를 말함. 단 $r \leq 470$)이므로 해독하기가 거의 불가능하며, CIST를 HDES1로 암호화하여 $S = E_{SK}(CIST)$가 생성된다. 따라서 인가된 송·수신자만이 SK(session key)를 소유하고 있으므로 S를 복호화($CIST = D_{SK}(S)$)하고 CIST를 이용하여 CP를 복호화해야만 원문을 생성시킬 수 있다. LZWHDES1는 CIST만을 암호화하므로 일반적으로 원문 전체를 압축하고 암호화하는 방법 보다 수행속도가 훨씬 빠르다는 이점이 있다. 또한 LZWHDES1는 관용키 암호시스템에 해당되며 데이터통신망에서 인감도장 역할을 하는 디지탈서명 중에서 중재자 서명시스템을 구현할 수 있다는 것이다. 그림 9.은 LZWHDES1을 이용하여 디지탈 서명을 구현할 수 있는 간단한 수행절차이다.

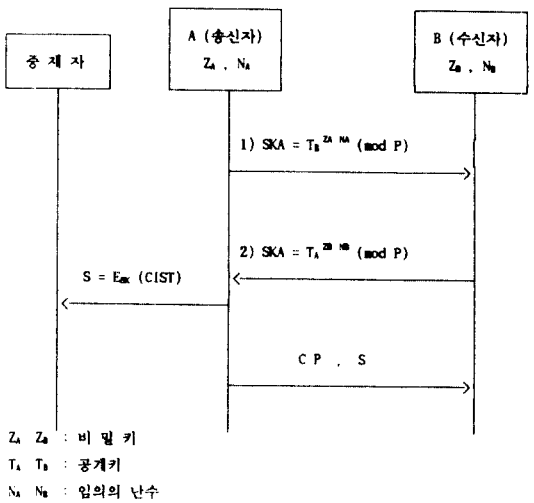


그림 9. LZWHDES1를 이용한 디지탈서명 시스템
Fig. 9. The arbitrator digital signature system using LZWHDESI

그림 9.에서 보인 바와 같이 Z_A와 Z_B는 유한체 GF(P) 상의 임의의 원소이며, T_A는 $T_A = g^{ZA} \pmod P$의 관계에 있으며, T_B는 $T_B = g^{ZB} \pmod P$이다. 여기서

g는 유한체 GF(P)상의 원시 원소이다. 1)과 2)의 통신으로 수신자 B는 다음과 같이 SK를 얻게되고

$$SK = SK_A^{NB} = T_B^{ZA\ NA\ NB} = g^{ZA\ ZB\ NA\ NB} \pmod P \quad (3.12)$$

송신자A도 다음과 같이 SK를 얻게된다.

$$SK = SK_B^{NA} = T_A^{ZB\ NB\ NA} = g^{ZA\ ZB\ NA\ AB} \pmod P \quad (3.13)$$

따라서 송·수신자만이 동일한 SK를 얻게되며 중재자는 SK를 갖고 있지 않기 때문에 원문을 변경시킬 수 없으며, 수신자는 SK를 갖고 있으므로 CIST를 생성하여 원문을 수정할 수는 있으나 송신자와 수신자간의 분쟁시에는 송신자가 보낸 S가 중재자에게 등록되어 있으므로 수신자에 대한 부인봉쇄를 할 수 있다. 또한 송신자 서명(S)이 바로 생성되기 때문에 서명생성을 위한 사전통신이 필요하지 않아 통신 효율을 향상시킬 수 있는 특징을 LZWHDES1는 가지고 있다.

V. 분석 및 고찰

본 논문에서 사용된 표본은 완성형 한글에 있어서 신문, 논문, 소설, 잡지 등에서 발췌한 100개의 표본 중에서 객관성이 있고 크기가 각각 다른 표본 8개만 선택하였으며, 분석에 있어서는 압축율과 수행시간 그리고 일반적인 암호 강도의 척도인 U.D.에 대해서 분석했다. DES-like 암호시스템에서의 암호 강도의 분석인 Differential Cryptanalysis에 대한 일부 반응 방안으로 SAC과 상관계수에 만족하는 S-box 설계에 대하여 분석하였으며, 압축에 대한 비교대상은 LZW를 비교 대상으로하여 LZWH2와 LZWH4를 비교하였으며, 암호에 대한 비교대상으로는 DES를 기준으로 HDES와 HDES1을 비교하였으며, 압축과 암호 결합시스템에 대한 비교대상으로는 먼저 LZW로 압축하고 압축열 전체를 DES로 암호화하는 방법인 LZWDES[9]를 기준으로 LZWHDES와 LZWHDES1을 각각 비교하였다.

표 4.은 LZW을 기준으로 LZWH2와 LZWH4에 대한 압축율을 분석한 것이다.

평균 압축율에 대한 분석결과 LZWH2는 LZW보다 0.41 만큼 증가하였으며, LZWH4는 LZW보다 0.27 만큼 증가했다. 그러나 LZWH4는 LZWH2 보다 0.14 만큼 떨어짐을 보였다. 그 이유는 LZWH4을 수행하면 초기 스트링테이블을 압축한 CIST가 추가로 생성되므로 압축율이 다소 떨어짐이 보였다. 표 5.은 LZW와 LZWH2 그리고 LZWH4에 대한 압축 및 복호에 대한 수행 시간을 분석한 것이다.

표 4. LZW, LZWH2 그리고 LZWH4에 대한 압축율
Table. 4. The compression ratio for LZW, LZWH2 and LZWH4.

표 본	원문 크기	영문, 한자 혼합율	LZW	LZWH2	LZWH4
SAMP1	1139	3	1.36	1.64	1.33
SAMP2	4644	7	3.25	4.05	3.50
SAMP3	11949	5	1.83	2.23	2.11
SAMP4	22006	2	2.08	2.39	2.25
SAMP5	36637	4	1.94	2.29	2.27
SAMP6	40309	3	2.07	2.45	2.43
SAMP7	49913	2	2.02	2.41	2.39
SAMP8	64262	3	2.07	2.45	2.47
평균값		3.6	2.07	2.48	2.34

표 5. 압축에 대한 수행시간
Table. 5. Processing times for the compression.

단위 : μsec/byte

	LZW	LZWH2	LZWH4
SAMP1	877.96	877.96	877.96
SAMP2	215.33	215.33	215.33
SAMP3	83.69	83.69	140.07
SAMP4	45.44	45.44	61.32
SAMP5	54.59	81.88	93.18
SAMP6	49.61	49.61	75.23
SAMP7	40.07	60.10	80.17
SAMP8	46.68	62.24	73.39
평균값	182.12	184.53	202.08

표 5.에서 보인 바와 같이 SAMP1과 SAMP2는 원문의 크기가 작으므로 수행시간이 거의 일정하였으며 LZWH2와 LZWH4가 LZW보다 평균 수행시간이 증가함을 보였다. 그 이유는 LZWH2와 LZWH4에 완성형 한글에 대한 처리 알고리즘이 부과되기 때문이다. 따라서 압축율과 처리시간은 서로간에 상반됨을 보였다. 그러나 LZWH4는 암호 알고리즘과 효

과적으로 결합하므로써 오히려 실시간 통신환경에 사용할 경우에는 수행시간을 훨씬 단축시킬 수가 있다. S-box 설계에 있어서는 S-box내의 엔트리를 랜덤하게 배열하여 Differential Cryptanalysis에 대한 일부 대응방안으로 SAC과 상관계수의 조건에 만족되는 S-box를 S1~S8에서 S1~S16로 증가시켰다. 표 6.과 표 7.은 설계된 HDES내의 S-box에 있어서의 SAC과 상관계수에 관한 분석이다.

S-box에 대한 SAC조건은 표 6.에서 보는 바와 같이 $P_{i,j}$로 평가하는데 $P_{i,j}$가 0.5에 접근할수록 S-box가 잘 설계된 것이며, 상관계수에 대한 평가는 표 7.에서 보인 바와 같이 $\rho_{i,j}$로 평가하는데 $\rho_{i,j}$가 0에 접근할수록 S-box가 잘 설계된 것이다. 표 8.는 HDES1이 DES나 HDES보다 U.D.에 있어서 증가됨을 보였으며 HDES1이 DES나 HDES보다 암호 강도가 향상됨을 보였다. 표 9.은 LZW를 DES에 직접 연결한 LZWDES, LZWH2를 HDES에 직접 연결한 LZWHDES 그리고 LZWH4와 HDES1을 효과적으로 결합한 LZWHDES1의 수행시간을 분석한 것이며, LZWDES에 사용된 키는 8바이트로 SK=HANSEUNG을 사용하였으며, LZWHDES와 LZWHDES1

표 6. DES, HDES 그리고 HDES1의 $P_{i,j}$
Table. 6. The $P_{i,j}$ for DES, HDES and HDES1

BOX	DES	HDES	HDES1
S1	0.620	0.505	0.500
S2	0.633	0.508	0.508
S3	0.661	0.514	0.497
S4	0.165	0.512	0.508
S5	0.633	0.515	0.500
S6	0.651	0.524	0.508
S7	0.656	0.518	0.495
S8	0.625	0.497	0.508
S9			0.505
S10			0.500
S11			0.505
S12			0.508
S13			0.508
S14			0.500
S15			0.497
S16			0.495

표 7. DES, HDES 그리고 HDES1에 대한 $\rho_{i,j}$
Table. 7. The $\rho_{i,j}$ for DES, HDES and HDES1

BOX	DES	HDES	HDES1
S1	−0.195	−0.014	−0.048
S2	−0.188	−0.025	−0.030
S3	−0.165	−0.033	−0.049
S4	−0.232	−0.045	−0.040
S5	−0.184	−0.016	−0.035
S6	−0.183	−0.024	−0.037
S7	−0.153	−0.027	−0.045
S8	−0.176	−0.020	−0.034
S9			−0.039
S10			−0.014
S11			−0.043
S12			−0.033
S13			−0.036
S14			−0.028
S15			−0.039
S16			−0.034

표 8. DES, HDES 그리고 HDES1에 대한 U.D.
Table. 8. U.D. for DES, HDES and HDES1

샘 플	U.D.		
	DES	HDES	HDES1
SAMP1	40.31	44.95	46.32
SAMP2	6.11	12.52	15.27
SAMP3	59.81	132.52	145.64
SAMP4	64.16	169.10	188.71
SAMP5	115.94	307.14	350.24
SAMP6	82.36	232.84	271.41
SAMP7	81.55	228.77	253.68
SAMP8	95.37	244.54	269.63
평 균	68.20	171.56	192.61

표 9. LZWDES, LZWHDES 그리고 LZWHDES1에 대한 수행시간
Table. 9. Processing times for LZWDES, LZWHDES and LZWHDES1.

단위 : μsec/byte

표 본	LZWDES	LZWHDES	LZWHDES1
	암 호	암 호	암 호
SAMP1	1755.92	1755.92	877.96
SAMP2	430.66	446.12	215.33
SAMP3	669.51	690.80	251.06
SAMP4	545.30	546.41	136.32
SAMP5	573.19	601.26	136.47
SAMP6	545.78	553.43	124.04
SAMP7	560.97	584.03	120.20
SAMP8	544.64	562.62	124.49
평 균	703.25	1717.85	248.23

에 사용되는 키는 16바이트로 SK=HANSEUNG-JOCHOSUN을 사용하였다.

LZWH1으로는 압축율이나 암호강도면에서 우수하기 때문에 컴퓨터에 정보를 저장·보관 시에는 우수하지만, 표 9.에서 보인 바와 같이 수행시간이 너무 지연되므로 실시간 통신 환경에 적합하지 못하다. 그러나 LZWHDES1는 LZWDES보다 수행시간이 바이트당 무려 455 μsec 만큼 짧아짐을 보였다. 그 이유는 LZWDES는 압축열 전체를 암호화하는 반면에 본 논문에서 설계한 암호코딩과 압축코딩의 결합시스템인 LZWHDES1는 초기 스트링테이블의 CIST만을 암호화하였기 때문에 전체적인 수행시간을 효과적으로 줄일 수 있었다. 따라서 LZWHDES1는 실시간 통신 환경에서 실용화 할 수 있다.

Ⅵ. 결 론

컴퓨터 통신망을 통하여 정보를 경제적이면서 효율적으로 안전하게 전송하기 위해서는 효율성, 기밀성 그리고 인증성이 요구된다. 이를 위해서 본 연구에서는 온 라인 전송에 있어서 디지탈서명을 위해 압축 코딩과 압축 코딩의 특성을 효율적으로 결합한 관용키 암호시스템인 LZWHDES1를 설계하였으며, LZWHDES1을 이용해 디지탈서명을 제안하였다. LZWHDES1는 압축알고리즘인 LZWH4와 암호알고리즘인 HDES1을 결합한 시스템이다. 일반적으로 통계적 기법의 압축알고리즘은 2패스 부호화 방법이지만, LZWH4는 적응적 사전방식인 LZW에 완성형 한글의 문자사용빈도에 따른 통계적 기법을 적용한 1패스 부호화 방법으로 디지탈 서명을 활용하기 위해 2개의 압축열이 생성되도록 설계하였다. LZWH4는 완성형 한글에 대한 압축율에 있어서 LZW 보다 0.27만큼 증가하나 LZWH2보다는 0.14만큼 감소됨을 보였으며, 수행시간도 LZWH와 LZWH2보다 증가하였음을 보였으나, LZWH4는 암호알고리즘과 효과적으로 결합하므로써 온 라인 전송에 사용할 경우에는 수행시간을 훨씬 단축시킬수 있다. HDES1는 DES를 확장하여 exhausitive 공격과 테이블 lookup 공격 그리고 time-memory trade off 공격 방법에 대응하기 위해서 키길이를 112비트로 증가시켰으며, differential crytanalysis에 대한 일부 대응방안으로 알고리즘을 수정하여 한 블럭당 96비트가 입력되게 하였으며, S-box내의 엔트리를 랜덤하게 배열하여 SAC와 상관계수조건에 만족하는 S-box를 S1-S8에서 S1 S16으로 증가시켰다. 따라서 HDES는 DES보다 U.D.가 증가함을 보였다. 또한 LZWHDES1는 LZWHDES2보다 수행시간이 바이트당 무려 455(μsec)만큼이나 짧아짐을 보여 실시간 통신환경에 적합함을 입증하였다. 앞으로의 연구과제는 LZWHDES1에 효율적인 키분배방식을 적용하여 LZWHDES1를 이용한 중재자 없는 디지탈 서명을 구현하고자 한다.

참 고 문 헌

1. T.Berson, "Network Security : The Parts of the Sum," Proc of Security and Privacy, pp. 2-9. May, 1989.
2. H.J. Baker and F.C.Piper, Cryptography and Coding, clarendon press, oxford, 1989.
3. A.Flat and A.Shamir, "How to prove yourself : Practical solutions to identification and signature problems," Proco. of Crypto'86, pp. 186-194. Spinger-Verlag, 1987.
4. T.A. Welch, "A Technique for High Performance Data Compression." IEEE Computer 17, 6, pp. 8-19, 1984.
5. E. Hiham and A.Shamir, "Differential Cryptanalysis of DES-like Cryposystem," Proc. of CRYPTOLOGY, Vol.4, No.1, 1991.
6. NBS, "Data Encryption Standard," FIPS Pub. 46, U.S, National Bureau of Standards, Washington DC, Jan. 1977.
7. Hamming, R.W, Coding and Imformation Theory, Prentice-Hall, Englewood Cliffs, 1988.
8. J.ziv, A.Lempel, "A Universal Algorithm for Sequential Data Compression," IEEE Trans. on Information Theory, Vol. IT-23, no.3, pp. 337-343, May, 1977.
9. 한승조, 이성호, 구인선, "완성형 한글에 있어서 압축과 암호결합에 대한 S/W의 설계 및 구현" 한국정보과학회 논문지, 8, 1993.(8월 게재예정)
10. Ziv,J. and Lempel,A., "Compression of Individual Sequences via Variable Rate Coding," IEEE Trans. Information Theory, Vol. IT-24, N0. 5, pp.5306-5317, Sept.1978.
11. Hidetoshi Yokoo, "An Improved Ziv-Lempel Coding Schme for Universial Source Coding" 일본전자통신학회논문지, Vol.J68-A, No.7, pp.

664-671, June, 1985.
12. D.E.Denning, Cryptography and Data Security, Addision-Wesley, 1983.
13. P.J. Denning, Computer under Attack, Addison-Wesley, 1990.
14. W.Diffie and M.E.Hellman "Exhaustive Cryptanalysis of the NBS Data Encryption Standard," IEEE, Vol.10, No.6, pp.74-84, 1977.
15. M.E.Hellman, "A Cryptanalytic Time Memory Tradeoff," IEEE Trans. of Info. theory, Vol. IT-26, No-4, pp.401, July, 1980.
16. A.F.Wester and S.E.Tavares, "On the Design for S-boxes," Proc. of CRYTO'85, Springer-Verlag, 1985.
17. E. Biham and A.Shamir, "Differential Cryptanalysis of the full 16-round DES," Proc. of CRYPTO'92, 1992.
18. 배영선, 한승조, "확장된 DES(HDES)암호알고리즘의 설계 및 구현," 대한전자학회 종합학술논문집, pp.15-18, 7, 1993.

본 논문은 1993년도 조선대학교 학술연구비의 지원을받아 연구되었음.

韓 承 朝(Seung Jo Han) 正會員
1980년 : 조선대학교 전자공학과 학사
1982년 : 조선대학교 대학원 전자공학과 석사
1994년 : 조선대학교 대학원 전자계산학과 박사
1984년 3월이후 ~ 현재 : 조선대학교 전자공학과 부교수
1986년 6월 ~ 1987년 3월 : Univ. of New Oleans 객원 교수
※주관심분야 : 정보이론, 시큐리티, 마이크로프로세서 등.

李 相 鎬(Sang Ho Lee) 正會員
1976년 : 숭실대학교 전자계산학과 졸업(공학사)
1981년 : 숭실대학교 대학원 전자계산학과 졸업(공학석사)
1989년 : 숭실대학교 대학원 전자계산학과 졸업(공학박사)
1976년 1월 ~ 1979년 5월 : 한국전력공사 전자계산소 근무
1992년 9월 ~ 1993년 8월 : 캐나다 UBC 컴퓨터 과학과 Post Doc
※주관심분야 : 프로토콜공학, 암호학, 시뮬레이션등

具 然 卨(Yeon Seol Koo) 正會員
1964년 : 청주대학교 상학과 졸업(상학사)
1975년 : 성균관대학교 경영대학원 전자자료처리학과 졸업(경영학 석사)
1981년 : 동국대학교 대학원 통계학과 졸업(이학석사)
1988년 : 광운대학교 전자계산학과 졸업(이학박사)
1979년 이후 : 충북대학교 전자계산소장 역임
한국정보과학회 이사, 전산교육연구회 위원장역임, 충청지부장 역임
현재 : 충북대학교 자연과학대학 장
한국 정보과학회 부회장
충북대학교 컴퓨터공학과 교수
※ 主關心分野 : 소프트웨어공학, 정보통신 알고리즘등