암호화법 (Encryption)

文 相 在
(正 會 員)
慶北大學校 電子工學科 副教授

Ⅰ. 머리말

군용 암호화법(encryption)은 민간용에 비하여 오랜 역사를 지니고, 기술적으로도 상당히 발전되어 왔다. 민간부분의 민감한 정보에 대한 보안의 필요성이 국가안보차원에서 절실해 짐에 따라 이 두 분야에서의 암호화 관련 정책은 서로 병합하는 추세이다. 다음의 사실이 이를 뒷받침한다. 미국의 NSA (national security agency)는 국방부의 기밀(classified) 정보에 대한 보안업무를 전담하도록 트루만 대통령에 의해 1952년 설치된 기구이다. 그러나 1984년 9월 17일자 레이건 대통령이 서명한 NSDD (national security decision directive) No. 145에 따라 모든 정부 부처의 민감한 정보에 대한 보안업무도 NSA가 담당하도록 하는 동시에 민간분야에 대해서는 지도역할을 하도록 그 기능이 확대되었다. 이는 미국 통신보안 정책에 큰 전환점이라 볼 수 있다.

중요한 군사적 정보들인 작전계획, 군사훈련, 군사적 능력, 보유 무기의 배치와 성능등이 사무자동시스템과 컴퓨터통신망에 의하여 관리 운용될 것이며 통신망을 통하여 관련 외부(행정부서 및 민간부분등)와 연결되므로, NSDD No. 145와 같은 통신보안 정책은 당연한 것이며, 모든 국가에서도 유사한 정책을 도입할 것으로 보여진다. 앞으로 NSA에서 인가할 암호화 알고리즘은 DES (data encryption standard)[1]의 경우와는 달리 정부의 통제하에 일체 공개되지 않는다. 그리고 NSDD No. 145에 따라서 1988년 1월 1일부터 기밀 및 민감한 정보를 암호화하는 정부용 통신보안 장비에 DES의 사용을 금지시켰다. 따라서 앞으로 국가마다 암호화 시스템의 개발을 위한 연구와 기술 축적이 더욱 절실히 요구될 것이다.

정보의 보안을 위하여 제한구역 설정, 자물쇠 사용, 경비자의 활용, 이해 집단의 지리적 분리등 물리적 혹은 개인적 보안을 채택하여 인가된 적임자만 기밀 혹은 민감한 정보에 접근할 수 있게 하는 방법을 사용할 수 있지만, 보다 효과적이고 실제적인 방법은 암호법(cryptography)을 적용하는 것이다. 경우에 따라서는 이 두 형태의 방법을 적절히 병용해야 한다.

암호법을 전송보안(TRANSEC, transmission securirty)과 혹은 컴퓨터 보안(COMPUSEC, computer security) 중 어떤 보안에 사용하느냐에 따라서 고찰 범위와 중점이 달라질 수 있다. 전송보안인 경우에는 주로 암호화법(encryption), 암호분석 방법과 암호비도(crypto-degree), 그리고 취약성(vulnerability) 등을 열거할 수 있을 것이며, 컴퓨터보안인 경우에는 상기 열거된 범위외에도 암호키이의 관리, 인증(authentication)방법, 통신망의 제어 및 프로토콜(protocol)의 수용, 데이타베이스 보안관리 그리고 소프트웨어와 firmware와의 결합등 보다 시스템적인 접근방식으로 암호법이 다루어져야 할 것이다. 이들 두 보안을 종합하여 일반적으로 통신보안(COMSEC, communication security)이라 한다. 통신보안은 통신의 소유나 연구에서 유추해 낼 수 있는 중요 정보를 비인가자의 접근을 방지하도록 고안된 모든 방법을 포함하나, 대부분의 사람들에게는 암호화의 방법이 곧 통신보안의 방법으로 느끼는 이유는 암호법이 중추적 역할을 하기 때문일 것이다.

지면이 제한된 본고에서는 암호화법 가운데서 데이타의 암호법와 공용키이 암호법에 국한시켜서 문헌으로 공개된 알고리즘중에 대표적인 것을 소개하고자 한다. 이들을 대체(substitution)와 치환(permutation)을 병용하여 암호화하는 것과 정수(integer)나 다항

식 개념을 데이타에 도입하고 이들에게 지수승(exponentiation)을 취하여 적절히 암호화하는 것으로 대별할 수 있다. 전자에 속하는 대표적인 것이 LUCIFER[2]와 DES이고, Trapdoor Knapsack 방식,[3] RSA방식[4]과 SEEK[5] 방식이 후자에 속하는 대표적인 것이다. 지금까지는 데이타의 고속 암호화에는 DES가, 그리고 공용키이 암호화에는 RSA방식과 SEEK 방식이 채택 응용된다. 먼저 LUCIFER와 DES의 동작원리를 소개하고, 암호비도의 중요한 파라미터가 되는 심볼간 상호의존에 대하여 서로 비교한다. 이러한 특성의 고찰은 DES형태의 암호화 알고리즘의 개발에 도움이 될 것으로 보인다. 그리고 RSA 방식을 중심으로 공용키이 암호법에 대해서 알아보도록 한다. 이들 두 형태의 알고리즘의 내용을 알아보기 전에 미국의 암호화 알고리즘의 최근동향을 간략하게 소개한다.

Ⅱ. 암호화 알고리즘의 최근동향

NSDD No. 145 이전에 NSA는 각 군사 프로그램마다 요구하는 통신보안(COMSEC) 장비를 생산하였다. 이는 대부분이 서로 다른 알고리즘을 갖는 수많은 암호화기들이 기존 통신시스템에 추가되는 결과를 가져왔다. 암호문이 불법적으로 해독된다해도 채택된 알고리즘은 전체 정보의 일부에만 사용될 수 있어 도용되는 정보가 작고, 적의 입장에서 볼 때 엄청난 계산 장비와 노력으로 해독해야 하므로 통신시스템이 지극히 안전하나, 다음 3가지 결점이 있다. 먼저 많은 다른 형태의 알고리즘들을 사용하므로 대부분의 암호화 장비들간에 호환성이 없다. 다음으로는 다종 소량 제조로 생산단가를 절감할 수 있는 경제성이 없다. 그리고 통신보안 장비들은 각기 설계되고 인가를 받으므로, 현장 투입시까지 개발기간이 오래 걸린다.

DES의 책임을 지닌 NBS(national bureau of standards)는 NSA와 상반되는 접근방법을 택했다. 단일 암호화 알고리즘으로 DES만 채택한 것이다. NSDD No. 145 이후, NSA는 암호화 알고리즘의 안전을 충족시키면서, 상기한 3가지 결점을 보완하는 절충방법을 채택하여 6가지의 암호화 알고리즘을 개발하였다.[6] 이들은 미래 암호응용에서 계획된 85%에 사용될 것이다. 정보를 기밀정보는 type Ⅰ으로, 기밀은 아니지만 민감한 정보이면 이를 type Ⅱ로 분류한다. 정보의 형태별로 암호화 알고리즘이 다르고, 또한 동일 형태에서도 응용분야에 따라 서로 다른 암호화 알고리즘이 지정된다. 응용분야는 고속 데이타용, 컴퓨터망용 그리고 음성 및 저속 데이타용의 3가지 분야로 구분된다. 결과적으로 6가지 조합의 각 경우마다 암호화 알고리즘이 지정된다. NSA에 의해 이들 6가지 암호화 알고리즘을 실현시킨 모듈(module)들은 표 1과 같다.

표 1. 암호화 모듈

응용분야 / 정보형태	고속데이타	컴퓨터망	음성/저속 데이타
Type Ⅰ	Foresee	Tepache	Windster Indicator
Type Ⅱ	Brushstroke	Bulletproof	Edgeshot

NSA는 표 1의 모듈을 사용한 통신보안 장비들을 승인 및 사용처 인가를 담당한다. Type Ⅰ 정보를 위한 통신보안 장비의 사용허가는 정부 및 정부의 계약자들에만 국한되고, 엄격한 통제를 받는다. Type Ⅱ 정보를 취급하는 정부 및 그 계약자들은 Type Ⅱ를 위한 통신보안 장비를 사용하며, 특히 NSA는 민간부분에서도 승인하에 이 통신보안 장비를 사용하도록 적극 지도하고 있다. NSA의 모듈에 대한 통제와 사용처에 대한 것을 그림 1에 도시하였다.

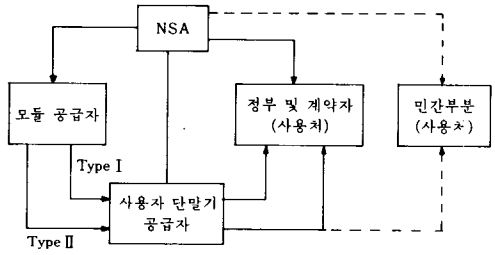

그림 1. 모듈에 대한 NSA의 통제도

모듈의 블록선도는 그림 2와 같다. 그 모듈들은 7개의 키이(key) 발생기와 한개의 키이관리기(key management module)를 내포하고 있다. 이 모듈들은 5볼트 전원용이며, TTL 입출력 장비와 호환성을 갖는다. Type Ⅰ의 모듈에 대한 공개된 사양(specs)은 표 2와 같다.[6] 그러나 NSA의 모든 모듈들의 알고리즘들은 통제하에 공개되지 않으므로, 키이 크기

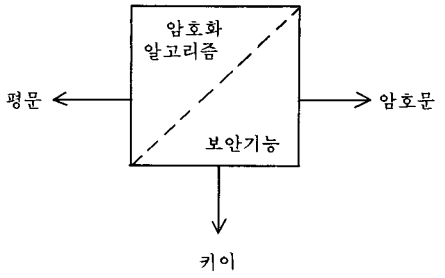

그림 2. 모듈의 블록선도

표 2. Type Ⅰ 모듈의 사양

모 듈 명	크 기	무 게	데이타율	암호모드	통신링크
Tepache	3″×4″×0.5″	6 OZ	10Mb/s	6	half duplex
Windster	2″×3.5″×0.5″	4OOZ	150Kb/s	9	full duplex
Foresee	TBD	TBD	20Mb/s	7	fulll duplex

조차 일반에게 알려져 있지 않다. 또한 이들의 국외 판매도 금지되어 있다.

지금까지 살펴본 바와 같이, NSA의 감독하에 정부와 민간 두 분야에서의 기밀 및 민감한 정보를 보호하기 위한 암호화 정책이 병합되어가고 있으며, 암호화법을 일체 공개하지 않으므로써 보안을 더 강화하였다. 다음은 암호법의 기본 개념과 대표적인 암호화법에 대하여 고찰한다.

Ⅲ. 암호법의 기본개념

1. 암호화 및 복호화

그림 3에서와 같이 암호화기는 평어(平語, plain word) M을 키이 K를 사용하여 암호어(cipher word) C로 변형한 후, 수신측으로 전송한다. 수신측에서는 키이 K로 암호어 C로 부터 복호하여 평어 M을 얻는다. 본 고에서는 이 과정의 수식적 표현을

$$C = E_K(M) \quad (1)$$
$$M = D_K(C) \quad (2)$$

로 표시한다. 여기서 M과 C는 주로 이진 비트열, 정수 혹은 다항식 형태로 정보를 대변하고, $E_k(\ )$/$D_k(\ )$는 채택된 암호화/복호화 과정을 나타내는 함수이다.

암호화 방법은 처리하는 형태에 따라 stream 암호화 방법과 block 암호화 방법으로 나눈다. Stream 방

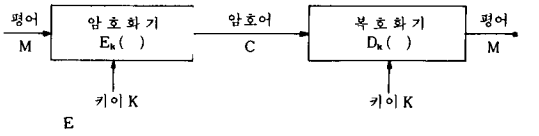

그림 3. 암호화 및 복호화 시스템의 블록선도

법은 문자 그대로 입력되는 정보비트들을 개별적으로 그리고 연속적으로 암호화시키는 것인데, 주로 키이 발생기에서 연속적으로 키이비트들을 출력시키고, 이 두 비트간에 mod-2를 취한다. 여기서 크기가 정해진 암호키이를 키이 발생기에 제공하여 키이비트열을 출력시킨다. 반면에 block 방법은 일정한 크기의 블록으로 정보비트들을 만들고, 이들을 동시에 암호화시키는 방법이다. DES가 대표적인 예이다. 응용하는 통신시스템에 적합하도록 이들을 선택 사용해야 하나, 일반적으로 채널오류비트의 영향이 확산되지 않고, 기존 통신시스템에 첨가시키기가 비교적 간단한 stream 방법이 많이 응용된다. 그림 4에 동기식 stream 암호화 방법을 도시하였다.

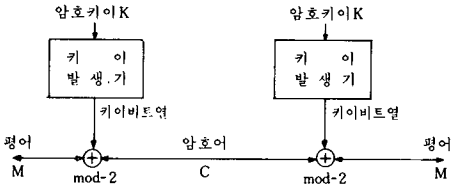

그림 4. Stream 암호화 방법

2. 암호분석

키이 K를 모르는 제 삼자가 분석을 통해 암호화법을 해독하는 것을 암호분석(cryptanalysis)이라 하고, 암호분석을 목적으로 하는 관련 행위를 attack이라 한다. 암호분석을 위해 암호문을 가로채려는 행위를 수동적(passive) attack이라 하고, 보다 과감하게 불법적으로 암호문을 변형 혹은 대체시켜 통신보안을 혼란시키는 행위를 능동적(active) attack이라 한다.

암호분석에 제공될 수 있는 자료에 따라서 ciphertext only attack, known plaintext attack, 그리고 chosen plaintext attack으로 구분한다.[7]

Ciphertext only attack은 암호문만 가지고 분석을

시도하는 것으로 실제로 자주 사용하는 행위이다. 평문에 관한 정보가 없으므로 예상되는 단어나 문귀 또는 사용빈도의 통계적 특성을 아는 문자등을 평문으로 활용한다. 세가지 가운데 가장 약한 위협이므로, 이 방법에 의해 분석되는 암호법은 안전성이 전적으로 결여되었다고 볼 수 있다.

Known plaintext attack은 상당한 양의 평문과 해당 암호문을 가지고 암호분석을 시도하는 것이다. 기밀 정보가 아니면 이전 통신한 내용이 대외비로 통제받지 않으므로 이러한 위협이 현실적으로 있을 수 있다. 암호법이 이 위협에 안전하다면 이전에 사용한 모든 정보의 보안을 유지할 부담이 없어진다. Known plaintext attack에서 시도되는 한가지 방법은 모든 종류의 키이를 적용시켜 보는 것인데 이 경우에 필요한 기억용량 또는 계산 부하가 막대하여 정보의 가치에 비추워 분석을 시도하고자 하는데 경제성이 전혀 없어야 할 것이다.

Chosen plaintext attack은 암호분석자가 임의로 수없이 평문을 선택할 수 있고, 결과의 암호문을 분석할 수 있는 여건을 갖춘 것으로, 실제로 이러한 경우가 발생되기는 어렵다.

일반적으로 암호분석에서는 분석자는 암호화 알고리즘을 알고 있다고 가정한다. 지적되었듯이 앞으로는 알고리즘도 공개되지 않으며, 실현시킨 회로소자로부터 판독이 되지 않도록 물리적 장치가 강구될 것으로 보인다.

3. 공용 키이 암호법의 기본 개념과 인증

모든 통신에서 상대방을 확인하는 인증(authentication) 문제는 매우 중요하다. 서류를 매체로 하고 직접적인 서명, 인장의 날인 혹은 지문의 수단으로 인증하는 방법이 통신시스템에서는 다른 형태로 구현되어져야 한다. 공용 키이 암호법(public key cryptography)이 인증방법을 구체적으로 실현시킨다.

공용 키이 암호법의 기본 개념과 인증방법은 다음과 같다. 어떤 통신시스템의 두 가입자를 A와 B로 나타내기로 한다.

가입자 A는 임의의 정보 M에 대하여

$$D_{dA}(E_{eA}(M))=E_{eA}(D_{dA}(M))=M \qquad (3)$$

를 만족하는 암호키이 eA와 복호키이 dA를 가지고 있다고 가정한다. B도 같은 성질의 암호키이 eB와 복호키이 dB를 가지고 있다고 가정한다. 공용 키이 암호법은 암호키이들(eA와 eB)을 공용하기 위하여 공개시키고, 복호키이들(dA와 dB)은 해당되는 본인들만 사용할 수 있도록 비밀로 한다.

A가 B를 확인하고자 할 때는, A는 렌덤비트열 X를 발생시켜 공개된 B의 암호키이 eB로 $E_{eB}(X)$ 시켜 B에게 보낸다. B는 이를 복호시켜 X를 얻는다. 즉 $D_{dB}(E_{eB}(X))=X$. 여기서 B가 아닌 제 삼자의 경우는 dB를 알지 못하므로 X로 복호하지 못한다. 다시 B는 공개된 A의 암호키이 eA로 $E_{eA}(X)$를 취하여 A에 전송하면 A는 자기만 알고 있는 dA를 사용, $D_{dA}(E_{eA}(X))=X$를 얻음으로 처음 렌덤비트열 X를 확인 한다. 따라서 A는 B를 인증하게 되고, 같은 방법으로 임의의 두 가입자간에 서로 인증할 수 있다. 이를 도시한 것이 그림 5이다. 이러한 공용 키이 암호법의 핵심부분은 식 (3)을 만족하는 암호화법의 개발인데 RSA방식 등 여러가지가 있다.[10]

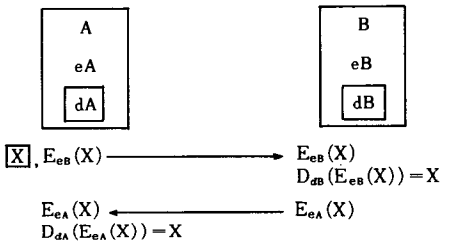

그림 5. 인증과정

공용 키이 암호법은 디지틀 서명(digital signature)의 기능도 갖는다. B에서 A의 디지틀 서명을 확인하는 방법을 도시한 것이 그림 6이다. 정보를 M이라 두면 A는 비밀의 dA를 사용하여 $D_{dA}(M)$를 취하면 일종의 암호문이 되는데 $D_{dA}(\ )$가 디지틀 서명에 해당된다. A는 공개된 eB로 다시 $E_{eB}(D_{dA}(M))$를 구하여 B에게 보내면 B는 식 (3)에 의하여 $D_{dB}(E_{eB}(D_{dA}(M)))=D_{dA}(M)$를 얻은 후에 오직 $E_{eA}(\ )$에 의해서만 $E_{eA}(D_{dA}(M))=M$를 얻을 수 있어 A가 서명

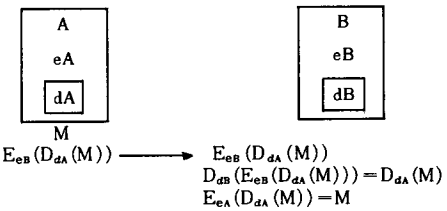

그림 6. 디지틀 서명 방법

한 것을 등가적으로 확인하게 된다.

Ⅳ. LUCIFER와 DES

LUCIFER와 DES는 블록으로 입력되는 자료비트에 대하여 대체와 치환을 여러번 병용하여 자료비트를 암호/복호화 하는 블록 암호/복호화 알고리즘으로 모두 미 IBM사에 의해서 개발되었다. 1970-71년에 LUCIFER가 개발되었고, LUCIFER를 근거로 이후 개발된 DES는 1977년 미 NBS(national bureau standard)에 의해서 표준 알고리즘으로 채택되었다.[1,2]

DES에 대해 암호화 과정이(복호화 과정도 유사하여 암호화 과정만 고려) 단순하나, 키이크기가 두배인 LUCIFER는 DES의 前身이라고도 일컬어진다. 이 절에서는 이 두 알고리즘의 구성을 서로 비교하여 대체와 치환을 병용하는 암호화 방법의 원리를 살펴본다. 그리고 암호의 불법적 분석을 방지하는데 중요한 요소가 되는 심볼간의 상호의존에 대해서도 비교해 본다.

1. 알고리즘의 구성비교

DES에 입력되는 데이타비트와 사용되는 키이비트의 블록크기는 64비트인 반면에 LUCIFER는 128비트이다. 앞으로 정보비트 블록을 평어(平語) 그리고 암호비트 블록을 암호어라 칭한다. 이들의 자세한 암호화 과정은 각각 부록 A와 B에 설명되었다.

DES와 LUCIFER는 입력되는 평어에 대하여 동일 형태의 과정을 16라운드 반복하여 암호화하는데, 매 라운드에서 다음 라운드로 넘어갈 때, 前 라운드에서 얻은 최종 블록을 크기가 동일한 하단(lower) 부블록과 상단(upper) 부블록으로 나눈 후, 과정이 반복 수행된다. 이러한 16라운드의 이전과 이후에 DEC만 그림 1에 보는 바와 같이 각각 한 번 더 치환과정을 수행한다.

매 라운드의 암호화 과정은 주로 대체와 치환으로 구성되는데 DES와 LUCIFER는 그 방법이 서로 다르다. 이 방법에 따라 정보 비트들 사이에, 그리고 정보비트와 열쇠비트 사이에 일어나는 혼합의 정도가 달라져 알고리즘의 안전에 영향이 온다. 매 라운드의 암호화 과정에서 LUCIFER는 비선형 대체상자(S-box)를 2개 가지고 있는 반면에, 이것의 4배 크기에 해당되는 대체상자를 DES는 8개 사용함으로써 대체에 의한 혼합성을 매우 강화하였다. 그리고 치환과정을 살펴보면, LUCIFER는 8비트 크기의 부블록으로 나누어, 각 부블록내에서 치환을 수행하지만, DES에서는 32비트 크기의 부블록으로 나누어 수행되므로, 상호간에 혼합될 수 있는 영역이 크게 넓혀졌다. DES는 LUCIFER에 비해서 키이크기를 반으로 축소시킨 대신에 대체와 치환과정을 보다 복잡하게 설계해서 안전을 강화하였다고 볼 수 있다.

암호분석을 위한 키이 exhaustive attack에서 LUCIFER는 DES에 비하여 2^{64}배 더 시도를 해야 한다. 그러면 LUCIFER가 더 안전하다고 할 수 있으나, DES에 비해 블록당 암호화 시간이 짧다는 것에 유의해야 한다. 일반적으로 암호화 알고리즘의 안전은 키이의 블록크기가 클수록 강화되고 암호화 처리속도가 고속일수록 약화되므로 이 두 요소를 적절히 절충해야 한다.

2. 심볼간 상호의존

DES나 LUCIFER에서 매 라운드마다 출력되는 비트를 평어와 키이의 함수로 간주할 수 있다. 만약 출력되는 한 비트가 평어의 모든 비트의 영향을 받아 변환하였다면, 이 출력비트는 평어와 상호의존한다고 말한다. 같은 의미로 출력비트는 키이와 상호의존할 수 있다. 주로 암호어와 평어 간의 상호의존, 그리고 암호어와 키이간의 상호의존으로 나누고 고찰하는데, 각 경우 암호어의 비트중 상호의존하는 비율을 백분율(%)로 표시하고, DES나 LUCIFER의 매 라운드마다 이러한 비율을 고찰하여 암호화 방법의 복잡성을 서로 비교한다.

이러한 상호의존에 관한 성질이 LUCIFER와 DES에 대해서 연구되었다.[8,9] 본 고에서는 이들의 연구결과를 아래에 소개한다.

DES나 LUCIFER의 매 라운드에서 평어는 하단 부블록과 상단 부블록으로 나누어지는데 이들을 각각 L_i와 U_i로 표시토록 한다. 여기서 첨자 i는 라운드의 수를 나타내는 것으로 $i=0, 1,2,\cdots,16$이고 L_0와 U_0는 초기 평어의 하단 부블록과 상단 부블록이다. 라운드별 LUCIFER와 DES의 암호어가 갖는 평어에 대한 상호의존 비율은 표 3과 같다. 상호의존 비율이 100%가 되기 위해서는 두 알고리즘 모두 5라운드가 필요함을 알 수 있다. 5라운드 이후는 100%의 상호의존 비율이 유지된다. 표 3에서 보면, LUCIFER에서 제 2라운드의 출력 암호어의 하단 부블록 L_2의 1.5%에 해당되는 비트들이 평어의 하단 부블록 L_0에 상호의존하고, 6.2%에 해당되는 비트들은 평어의 상단 부블록 U_0에 상호의존하는

표 3. 평어에 대한 암호어의 상호의존 비율

Round i	LUCIFER			DES		
	L_i vs. L_0	L_i vs. U_0 U_i vs. L_0	U_i vs. U_0	L_i vs. L_0	L_i vs. U_0 U_i vs. L_0	U_i vs. U_0
1	0.00	1.56	6.25	0.00	3.13	18.75
2	1.56	6.25	50.78	3.13	18.75	87.60
3	6.25	50.78	100.00	18.75	87.60	100.00
4	50.78	100.00	100.00	87.60	100.00	100.00
5	100.00	100.00	100.00	100.00	100.00	100.00

반면에 DES의 경우는 그 비율이 두배에 가깝다. LUCIFER에 비하여 DES의 암호화 방법이 더 복잡하여 대수적 암호분석에 더 강하다.

같은 방법으로 연구한, 암호어가 키이에 대한 상호의존 비율은 표 4와 같다. DES의 5라운드수에 비해서 LUCIFER는 6 번째 라운드에 이르러서야 키이에 대한 상호의존 비율이 100%가 된다.

표 4. 키이에 대한 암호어의 상호의존 비율

Round i	LUCIFER		DES	
	L_i vs. key	U_i vs. key	L_i vs. key	U_i vs. key
1	0.00	1.60	0.00	10.71
2	1.60	13.00	10.71	79.02
3	13.00	56.00	79.02	96.43
4	56.00	95.00	96.43	100.00
5	95.00	100.00	100.00	100.00
6	100.00	100.00	100.00	100.00

표 3과 4에서 알 수 있듯이 제 6라운드 이후는 암호어의 각 비트는 평어와 키이의 모든 비트와 상호의존 즉 복잡한 함수로 영향을 받는다. 그리고 계속해서 제 16라운드까지 암호화가 수행되어 대수적 암호분석이 불가능한 것으로 여겨지고 있다. 그러나 암호 키이를 알고 있으면 키이 K_i를 역순으로 하여 한 번 더 암호화 시키면 입력된 암호어가 평어로 해독되어 출력된다.

V. 공용 키이 암호법

앞 절에서 설명된 LUCIFER나 DES를 사용한 암호시스템에서는 송신자와 수신자는 동일한 키이를 사용하여 암호화와 복호화를 한다. 여기서는 키이를 안전하게 분배시키는 문제와 상대방을 확인하는 인증문제가 언급되지 않았지만 필히 해결되야 한다. 다음에 소개될 공용 키이 암호법은 Ⅲ.3절에서 설명된 바와 같이 이러한 문제가 해결될 뿐아니라, 평문을 암호화하는 데도 사용할 수 있다. 그러나 공용 키이 암호법의 처리속도는 LUCIFER나 DES에 비해 늦다.[10] 일반적으로 일정한 기간동안 사용한 키이(session key)를 암호화하여 분배할 때는 공용키이 암호법을 이용하고, 이들 키이를 사용하여 평문을 암호화할 때는 DES와 같은 고속 암호화기를 이용한다.

대표적인 주요 공용 키이 암호방식으로는 머리말에서 열거된 것외에도 Diffie-Hellman 방식.[7] McEliece의 공용 키이 암호법[11] 그리고 Pohlig-Hellman방식[12]을 들 수 있다. 이들 중에서 주로 채택 사용되고 있는 것은 RSA방식과 SEEK방식이다. 이 SEEK 방식은 Diffie-Hellman 방식과 Pohlig-Hellman방식을 혼합한 것이다. 본 절에서는 이 두 방식에 대해서만 알아보기로 한다.

1. RSA 방식

R. Rivest, A. Shamir와 L. Adleman에 의해서 고안된 RSA방식은[4] 정보 데이타를 整數로 나타내고, 이들 정수를 modulo 연산으로 멱승(exponentiation) 하여 다른 정수로 암호화하는 방법으로 DES와는 판이하다. 이때 멱지수(exponent) e와 연산을 위해 mod n가 필요한데 (e, n)를 암호화 키이라 한다. 같은 계산방법으로 복호할 때 사용되는 멱지수 d와 mod n의 (d, n)를 복호화 키이라 한다. 예로서 blank =00, A=01, B=02, ··, Z=26으로 정수화 하면 RSA ALGORITHM은 1819 0100 0112 0715 1809 2008 1300으로 정수화된다. 여기서 (e, n) = (1223, 2867) 이면

$$1819^{1223} \equiv 2756 \bmod 2867$$

이 되어 1819가 2756으로 암호화 되는 것이다. 그리고 (e, n) = (1223, 2867)에 대한 복호화 키이는 (d, n) = (167, 2867)이 되는데, 다음과 같이 복호가 된다.

$$2756^{167} \bmod 2867 = 1819$$

RSA방식에서 사용자는 암호화 키이 (e, n)를 공개하여 누구든지 공용할 수 있도록 하는 대신, 복호화 키이(d, n)를 비밀로 하여 본인만 복호할 수 있도록 한다. 따라서 암호화를 위한 키이의 분배문제가 자동

적으로 해결된다. 이러한 시스템이 수용되기 위해서 키이인 (e, n) 와 (d, n) 의 발생은 간단해야 하나, 공개된 (e, n) 와 암호문 자료로 부터 (d, n) 를 구하는 것 혹은 평문을 얻는 것이 지극히 어려워야 할 것이다. 아래의 설명에서 알 수 있듯이 RSA 방식은 이러한 성질을 만족한다.

암호화 원리를 설명하기 전에 우선 키이의 발생방법을 알아본다. Modulo 연산에서 사용할 n는 두 素數 (prime) 의 곱으로 구하는데, 암호분석 (n를 소인수분해 (factoring) 하여 두 소수를 알아냄) 에 대비하여 매우 큰 두 소수 p와 q를 임의로 선택해서

$$n=p\cdot q \qquad (4)$$

를 얻는다. 0보다 크고 n보다 작은 정수이면서 n와 서로 素인 것의 개수를 n의 Euler 함수 $\phi(n)$로 정의하는데, 두 소수의 곱인 경우는 쉽게

$$\phi(n)=(p-1)\ (q-1) \qquad (5)$$

임을 알 수 있다. $\phi(n)$를 구한 후에 {3,4,5,…, $\phi(n)-1$}중에서 $\phi(n)$와 서로 素인 임의의 정수를 선택하면 암호화 키이 e가 될 수 있다. 여기서 (e, n) 를 공개한다. 정해진 (e, n) 에 대한 복호화 키이 (d, n) 의 d는 mod $\phi(n)$에서 e의 역수

$$d=e^{-1} \bmod \phi(n) \qquad (6)$$

가 되야 한다. d는 사용자만 아는 비밀 키이이다. 식 (6)의 역수계산은 Euclid 알고리즘[13, pp315 & pp523]으로 쉽게 얻을 수 있어, 지금까지 설명된 바와 같이 키이의 발생은 간단한 편이다.

그러나 (e, n) 를 알고 d를 구하기는 매우 어렵다. 식 (6)의 $\phi(n)$를 알아야 하는데 이는 $\phi(n)=(p-1)\cdot(q-1)$ 이므로 n를 소인수분해하여 p와 q를 알아야 하는 것과 같다. 임의의 아주 큰 정수를 소인수분해하는 것으로 整數論에서 매우 어려운 문제로 되어 있다. 따라서 RSA 방식은 이러한 성질을 이용하여 고안된 것이다. 문제는 어느정도 큰 정수이면 실제적으로 취급할 수 있으면서 암호분석으로 부터 안전하다고 할 수 있느냐는 것이다. n의 자리수가 200개 이상이면, 암호분석이 현실적으로 (계산시간과 사용될 컴퓨터 장비값) 수행될 수 없는 것으로 되어 있다. RSA방식의 고안자들은 암호분석에 보다 안전하도록 하기 위하여, p와 q를 선택할 때 다음 사항들을 만족하도록 권고하고 있다.[4]

- 단지 몇자리수의 차이만 있도록 p와 q의 크기는 비슷해야 한다.
- (p−1)과 (q−1)는 각기 큰 소수 p′와 q′를 갖도록 한다.
- (p−1)과 (q−1)의 최대공약수는 작아야 한다.

다음은 암호화 원리를 살펴본다. Euler 정리로 부터 식 (5)와 같이 Euler 함수 $\phi(n)$가 주어지면 {0,1,2,3,…, n−1} 중의 어떠한 정수 a에 대하여

$$a^{k\phi(n)+1}=a \bmod n \qquad (7)$$

이 만족된다. 여기서 k는 임의의 정수이다. 정보 (평어나 키이) 를 n보다 작은 값으로 정수화하여 이를 M로 두면, 해당 암호어는

$$C=M^e \bmod n \qquad (8)$$

이다. 식 (6)을 만족하는 복호화 키이 d를 사용하면, 식 (7)식 성질에 의해

$$C^d=(M^e)^d=M^{k\phi(n)+1}=M \bmod n \qquad (9)$$

으로 복호된다.

식 (9)에서 알 수 있듯이 RSA 방식은

$$D_d(E_e(M))=E_e(D_d(M))=M \qquad (10)$$

인 식 (3)의 성질을 만족함으로, Ⅲ.3 절에서 설명한 인증과 디지틀 서명 기능을 가지게 된다.

2. SEEK 방식

SEEK (secure electronic exchange of keys) 방식도 데이타를 정수로 표현시키고, 이들 정수를 유한체 (finite field) GF(p) 에서 멱승하여 암호화하는 공용 키이 암호법이다. 여기서 p는 매우 큰 소수이다. 이 방식은 Diffie-Hellman 방식과 Pohlig-Hellman 방식을 혼합한 것으로 미 Cylink사에 의해서 firmware로 구현되어 시판된다.[5,14]

SEEK방식은 크게 두 부분으로 그 과정을 구분하여 설명할 수 있다. 첫째 과정은 두 교신자 A와 B 사이에 사용할 비밀 키이를 먼저 설정하는 것이고, 둘째 과정은 이 키이를 사용하여 정보를 교환하는 것이다.

이 방식이 동작되기 위해서는 먼저 임의의 정수 a와 매우 큰 소수 p (p>a) 를 선택하여 사용자들에게 공급한다. 이때 a와 p가 공개되더라도 이 방식의 안전에는 무관하나, 비밀로 취급되면 더 바람직할 것이다. 사용자 A와 B는 각자 p보다 작은 임의의 비밀 정수 X(A)와 X(B)를 발생시킨다. A는 $a^{X(A)} \bmod p$

를 계산하여 이를 B에게, B는 $a^{X(B)} \bmod p$를 계산하여 이를 A에게 각각 보낸다. 전송시 이들 값이 공개 되어도 무방하다. A와 B는 각각 수신된 값을 본인의 비밀 숫자만큼 멱승시킨다. 즉, A는 $(a^{X(B)})^{X(A)} \bmod p$그리고 B는 $(a^{X(A)})^{X(B)} \bmod p$ 를 계산한다. A와 B는 비밀 숫자인

$$Z(A, B) = (a^{X(A)})^{X(B)} \bmod p \qquad (11)$$

단 Z(A, B) = 홀수(짝수이면 최하위 비트 바꿈)

를 설정한다. p가 소수이므로 $\phi(p) = p-1$이다. A와 B는 정보 데이타 M을 암호화할 때는 Z(A, B)를 공통키이로 하여

$$C = M^{Z(A,B)} \bmod p \qquad (12)$$

를 취한다. 복호하기 위해서는 각자

$$Y(A, B) = Z^{-1}(A, B) \bmod \phi(p) \qquad (13)$$

를 계산하여 사용한다. 즉 Euler 정리에 의해서

$$C^{Y(A,B)} = M^{k\phi(P)+1} \bmod p = M \bmod p \qquad (14)$$

이다. 여기서 k는 임의의 정수가 된다.

만약 B가 아닌 제 삼자 T가 불법으로 접근하여 A와 통신할 때, A는 Z(A, B) 키이를 사용하나, T는 X(B)를 몰라 다른 키이를 가지게 되어 서로 정상적인 정보교환이 되지 않는다. A는 B가 아님을 알 수 있다. 이 방식의 안전은 $a^{X(B)}$ 혹은 $M^{Z(A,B)}$에서 X(B) 혹은 Z(A, B)를 알아내는 분석과 관련한다. p가 매우 크면 이러한 암호분석은 아주 어려운 문제로, RSA 방식에서의 큰 정수 n의 소인수분해와 상응하는 난제로 되어 있다.

Ⅵ. 맺음말

암호법은 통신보안을 위한 효과적이고 그리고 실제적인 수단으로 일찌기 군사용으로 사용되어 왔다. 근래에 와서 민간부분의 민감한 정보에 대한 보안의 필요성이 국가 안보차원에서 절실해 짐에 따라, 민간부분의 통신보안과 군사적 통신보안에 관한 정책이 국가적 차원에서 병합되어 깊게 다루어지고 있는 추세이다.

본 고에서는 이러한 최근 동향을 미국의 경우를 들어 간략히 언급하였으며, 문헌을 통해 공개된 암호화 알고리즘 가운데 대표적인 것들을 살펴보았다. 대체와 치환을 병용하여 암호화하는 대표적인 관용(conventional) 암호화 방법인 LUCIFER와 DES의 알고리즘을 소개하고, 이들의 심볼간 상호의존에 관해 알아 보았다. 그리고 공용 키이 암호법 가운데 실제로 수용되고 있는 RSA방식과 SEEK방식에 대해서도 알아 보았다.

앞으로 자국에서 개발된 암호화 알고리즘이 공개된 DES의 경우와는 달리 통제하에 일체 공개되지 않을 뿐아니라, 타국으로의 수출도 금지되는 것이 국제적인 추세이다. 따라서 국가마다 보다 안전한 암호화 시스템의 개발을 위한 연구와 기술 축적이 더 절실히 요구될 것이다.

부록 A : DES

그림 1 은 DES 암호화 과정의 흐름도이다. 입력(input)과 키이 K_i, i=1, 2, …, 16의 크기는 64비트이다. 키이 발생과정부터 자세히 살펴본다.

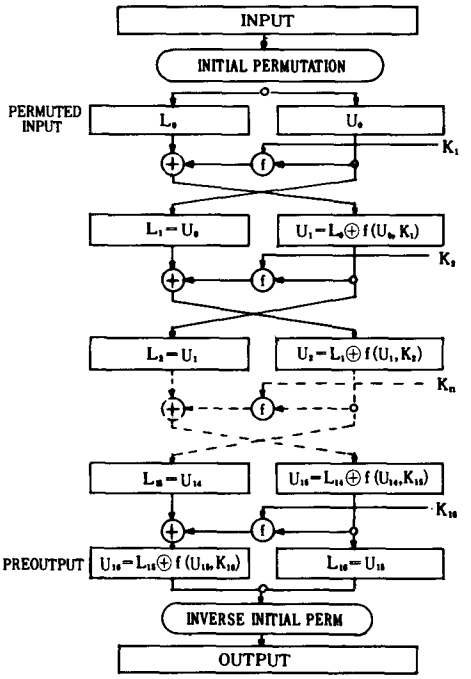

그림 1. DES 암호화 과정 흐름도

1. 키이 발생

각 라운드에 각기 사용될 16개의 키이 K_i, i=1, 2, …, 16의 생성과정을 흐름도로 나타낸 것이 그림 2 이다.

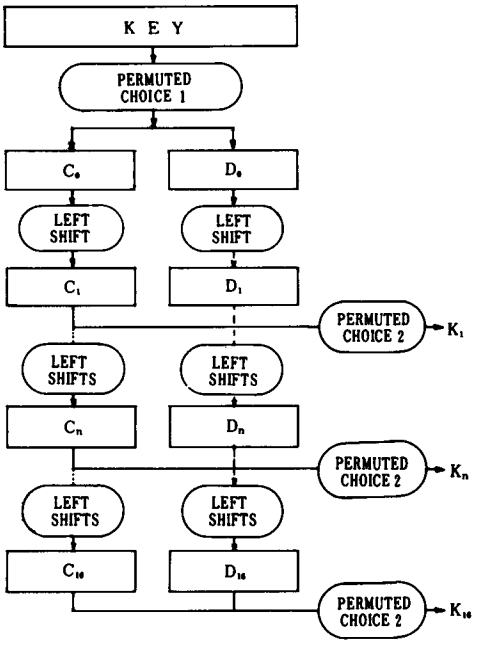

그림 2. 키이 생성과정 흐름도

먼저 표 1 의 permuted choice 1 에 따라서 64비트의 원 키이에서 56비트만(패리티 비트 8 개는 제외) 선택되어 치환하게 된다. 이들을 좌우 28비트씩 C_0와 D_0로 양분한다. C_0와 D_0를 표 2 의 이동 스케줄의 반복순서 첫번째에 의해서 1 비트 좌향으로 순환이동시켜 C_1과 D_1를 얻는다. C_1과 D_1를 다시 병합한 56비트에서 표 3 의 permuted choice 2에 의하여 선택 치환시켜 48비트의 키이 K_1를 발생시킨다. i 번째 ($2 \leq i$ 16) 라운드를 위한 키이 K_i도 같은 방법으로 발생시킨다.

2. 초기치환 및 역치환

그림 1 에서의 초기치환(initial permutation)은 입력되는 평어의 64비트에 대한 비트 자리바꿈을 행하는 과정이며, 표 4 에서와 같이 58번째 비트를 첫째 비트자리에, 50번째 비트를 두번째 비트자리로… 방법으로 자리바꿈을 행한다. 그리고 그림 1 에서의 최종치환(inverse initial perm)은 초기치환의 역과정이며, 같은 방법으로 표 5 에 의하여 비트 자리바꿈을 행한다.

표 1. Permuted choice 1

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

표 2. Permuted choice 2

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

표 3. 좌향이동 스케줄

Iteration Number	Number of Left Shifts
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

표 4. 초기치환

58	52	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

표 5. 최종치환

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. 암호화 함수 f

그림 1에서 보면, 초기치환후 얻어진 64비트를 32비트씩 하단 부블록 L_0와 상단 부블록 U_0로 양분시다. 함수 f는 U_0와 키이 K_1를 입력으로 해서 32비트의 출력을 생성하고, 이 출력과 L_0가 비트별 mod-2 가산되어 U_1이 된다. L_1은 U_0를 그대로 사용한다. 나머지 라운드도 같은 방법으로 수행된다.

함수 f는 DES에서 가장 중요한 부분으로 비트확장 과정, mod-2 연산과정, 대체과정 및 치환과정으로 구성된다. 그림 3은 함수 f의 블록선도이다.

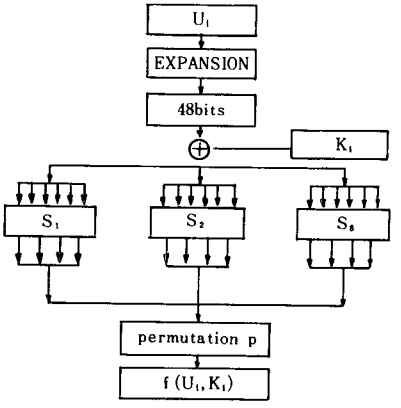

그림 3. 함수 f

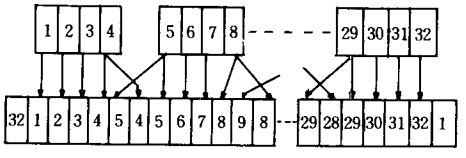

그림 4. 확장과정

그림 4는 그림 3의 확장(expansion) 과정을 자세히 나타낸 것이다. 그림 3에서와 같이 32비트의 U_i가 48비트로 확장된다. 확장된 48비트와 키이 K_i가 비트별 mod-2 가산되고, 얻어진 출력은 그림 3에서와 같이 6비트씩 나누어져 8개의 대체상자 S_i, i=1,2, …,6에 입력한다. 대체상자별 치환축소과정은 표 6과 같다. i번째 대체상자 S_i에서 1과 6번째 비트의 십진숫자(예로써 이진 10은 십진수 2)따라 표 6의 해당 S_i의 열을 선택하고, 2,3,4와 5번째 비트의 십진수에 따라서 해당 S_i의 행을 선택하여, 행과 열이 교차되는 십진수를 나타내는 4비트가 최종 치환축소된 출력이다.

그림 3의 마지막 단계인 치환은 표 7에 의하여 32비트에 대한 자리바꿈을 행하는 것이다.

표 6. 대체상자 S_i

Row	Column 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
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	S_1
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	
0	15	1	8	14	6	11	3	4	9	7	2	12	13	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	S_2
3	13	8	10	1	3	15	14	2	11	6	7	12	0	5	14	9	
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	S_3
3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12	
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	S_4
3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14	
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	S_5
3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3	
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	S_6
3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13	
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	S_7
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12	
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	S_8
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11	

표 7. 치환(permutation)

16	7	20	21
19	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

4. 복호과정

수신측에서의 복호는 암호어를 입력시키고 키이의 순서를 그 역순인 $K_{16}, K_{15}, \cdots, K_1$ 순서로 한번 더 암호화 하면 평어로 해독되어 복호된다.[1]

부록 B : LUCIFER[2]

LUCIFER의 평어, 암호어 그리고 키이의 크기는 공히 128비트이다. 이 LUCIFER는 같은 방법의 과정을 16번 반복 수행하며, 한 라운드에서 이루어지는 과정은 그림 5에 도시되었다. 평어 128비트는 64비트씩 하반부와 상반부로 양분된다. 매 라운드에서 상반부와 키이의 영향을 받아서 새로운 64비트를 구성한다. 이 변형된 64비트는 다음 라운드에 입력될 때는 하반부가 되고 상반부는 이전 라운드의 하반부를 그대로 사용한다.

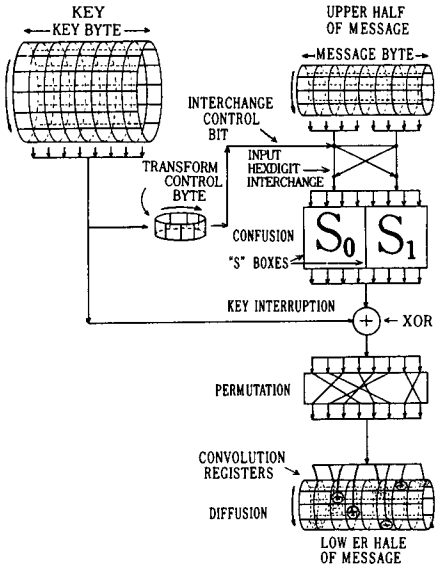

그림 5. CID(confusion interuption diffusion) 블록선도

그림 5에서와 같은 매 라운드의 암호화과정을 자세히 설명하면 다음과 같다. 128비트의 키이는 16바이트로 구분되고, 64비트의 평어 상반부는 8바이트로 구분되어 각기 그림 5의 원통과 같이 구성된다. 매 라운드마다 사용되는 키이 비트수는 평어의 상반부 비트수와 동일한 64비트이다. 즉 8바이트가 사용된다. 매라운드에서 마지막 바이트가 수행되고 나면 그 바이트는 회전하지 않고 정지하여 다음 라운드의 첫 바이트가 된다. 표 8은 매 라운드에 64비트 키이 발생을 위한 스케줄이다. 매 라운드마다 사용되는 키이의 첫 바이트는 변환조정바이트(tranform-control-byte)로 사용된다. 이들 8개의 비트들은 상호 교환조정비트(interchange-control-bit)가 되어서, 매 비트는 정보어 상반부가 어떤 방법으로 2개의 S 상자에 입력하는 지를 결정한다. 상호교환조정 비트가 "0"이면 정보어 상반부 바이트의 좌우 4비트의 두 십진값이 각각 S_0와 S_1의 입력값이 되고, "1"이면 두 십진값이 서로 바뀌어 들어간다.

표 8. 키이 바이트 사용 스케줄

	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

매 라운드마다 S_0와 S_1상자, 그리고 한 번의 치환과정이 8번 반복 수행하게 된다. 한 번 수행시 그 과정은 표 9에 의해 이루어지는데 그림 5에서와 같이 키이 바이트와 S상자 출력 바이트 간에는 비트별 mod-2 가산이 행하지고 그 결과가 치환상자에 입력된다.

치환과정을 거쳐 얻어진 바이트의 한 비트는 평어 하반부의 특정한 비트와 비트별 mod-2 가산을 하는

표 9. S 상자의 대체와 치환

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	

데 그 연산이 행해지는 방법은 그림 6 에 도시되어 있다. 예로써 첫 라운드에서는 평어 하반부를 구성하는 8 바이트에 대해 각 바이트마다 순서대로 5, 3, 2, 6, 7, 4, 1 그리고 0 번 비트만 치환과정에서 출력되는 8 바이트의 해당 비트들과 연산이 수행된다.

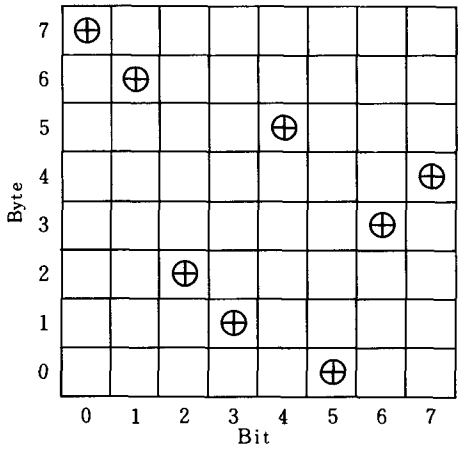

그림 6. 펼쳐진 콘벌루션 레지스터

상술한 바와 같은 매 라운드의 암호화 과정을 16 번 반복 수행되어 LUCIFER의 최종 출력 128비트가 얻어지게 된다. 수신측에서의 암호해독을 위한 복호는 암호화의 역순으로 그림 5의 CID 과정을 수행하면 된다.

參 考 文 獻

[1] National Bureau of Standards, Data Eneryption Standard, U.S. FIFP PUB 46, pp.1-18, 1977.

[2] A. Sorkin, "LUCIFER, a cryptographic algorithm," *Cryptologia,* vol. 8, no. 1, pp. 22-35, Jan. 1984.

[3] R. Merkle and M. Hellman, "Hiding information and signature in trapdoor knapsacks," *IEEE Trans. Infor. Theory,* vol. IT-24, pp. 525-530, Sep. 1980.

[4] R. Rivest, A. Shamir and L. Adleman, "A method for obtaining digital signitures and public-key crytosystems," *Comm. ACM,* vol. 21, no. 2, pp. 120-126, 1978.

[5] L. Neuwirth, "A comparison of four key distribution methods," Telecommunications, July 1986.

[6] L. Sanders, "Interoperability in Encrypted Communication Networks," Journal of Electronic Defense, pp. 95, Nov. 1987.

[7] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Trans. on Infor. Theory,* vol. IT-22, pp 644-654, Nov. 1976.

[8] 이훈재. 문상재, "LUCIFER와 DES에서의 심볼간 상호의존성에 관한 연구," 경북대학교 전자기술연구지, 제 8권, pp. 89-92, 1987. 8.

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

[10] M. Hellman, "An Overview of Public Key Cryptography," IEEE Comm. Society Magazine, pp. 24-32, Nov. 1978.

[11] R. McEliece, "A public key system based on algebra coding theory," JPL DSN Progress Rep., 1978.

[12] S. Pohlig and M. Hellman, "An improved algorithm for computing logarithms over GF(p) and its cryptographic significance," *IEEE Trans. Inform. Theory,* vol. IT-24, pp. 106-110, 1978.

[13] D. Knuth, *The art of Computer Programming;* vol. 2, Reading, MA, Addison-Wesley, 1969.

[14] C. Barney, "Cypher chip makes key distribution a snap," Electronics, pp. 30-31, August 7, 1986. *