The Journal of Korean
Institute of CALS/EC
Vol. 5, No. 1, June 2000

전자상거래를 위한 128비트 블록 암호 알고리즘의 구현

서장원*, 전문석**

An Implementation of 128bit Block Cipher Algorithm for Electronic Commerce

Jang-Won Suh, Moon-Seog Jun

Abstract

Recently, EC(Electronic Commerce) is increasing with high speed based on the expansion of Internet. EC which is done on the cyber space through Internet has strong point like independence from time and space. On the contrary, it also has weak point like security problem because anybody can access easily to the system due to open network attribute of Internet. Therefore, we need the solutions that protect the security problem for safe and useful EC activity. One of these solutions is the implementation of strong cipher algorithm.

NC(Nonpolynomial Complete) cipher algorithm proposed in this paper is good for the security and it overcome the limit of current 64bits cipher algorithm using 128bits key length for input, output and encryption key. Moreover, it is designed for the increase of calculation complexity and probability calculation by adapting more complex design for subkey generation regarded as one of important element effected to encryption.

The result of simulation by the comparison with other cipher algorithm for capacity evaluation of proposed NC cipher algorithm is that the speed of encryption and decryption is 7.63 Mbps per block and the speed of subkey generation is 2.42 μsec per blcok. So, prosed NC cipher algorithm is regarded as proper level for encryption. Furthermore, speed of subkey generation shows that NC cipher algorithm has the probability used to MAC(Message Authentication Code) and block implementation of Hash function.

Key Words : Security problem, Cipher algorithm, NC, Subkey

* 숭실대학교 대학원 컴퓨터학과
** 숭실대학교 컴퓨터학부 교수

1. 서 론

최근 들어 컴퓨터 통신 및 네트워크의 확산에 따른 활성화 추세에 따라 이를 기반으로 한 인터넷의 사용이 전 세계적으로 급증하고 있다. 인터넷의 용도는 지금까지의 단순 정보 교환이나 학술 연구를 주요 대상으로 한 제한적 정보 공유의 목적에서 네트워크를 근간으로 하여 인터넷을 하나의 거대한 마케팅의 대상으로 보고, 이를 상업적으로 이용하려는 무한적인 시도가 증가하고 있는 실정이다. 이런 인터넷을 이용한 상업화 추세가 급증하면서 웹을 활용한 인터넷 광고와 홍보, 전자신문, 전자출판 등을 비롯하여 소프트웨어, 생활용품, 가전제품 등과 같은 다양한 상품을 전시 및 판매하는 각종 인터넷 상점이 등장하였고, 또한, 많은 수의 인터넷 쇼핑몰들이 개장되고 있다. 그리고, 현재에는 동종 업종이나 업체뿐만 아니라 타 업종간의 통합 및 물류 서비스 체계까지도 활발히 진행되고 있는 실정이다. 이와 같은 상황 속에서 인터넷을 상업적으로 이용하기 위한 하나의 형태로 상품의 관련 정보에서 구매 및 결제에 이르기까지 네트워크를 통해 전자적으로 거래를 할 수 있는 전자상거래가 주목받고 있다[1].

전자상거래는 컴퓨터 기반의 디지털 데이터를 개방형 네트워크를 통하여 주고받는 특징을 가짐으로서, 거래 당사자간의 신속하고 원활한 거래가 이루어짐으로서 시간과 공간 절약을 도모할 수 있다는 장점이 있는 반면, 아직까지는 상호간의 신분 확인을 위한 시스템이나 절차가 용이하지 않기 때문에 이를 범죄에 이용하거나 또는 금융거래와 같은 형태에 이용하게 되는 경우에는 문제가 발생할 수 있다는 단점이 있다. 따라서, 이러한 전자상거래 상의 문제점을 해결하고 보다 안전하고 효율적으로 거래를 처리하기 위한 필수 선결요소 중에 하나가 보안 기술이다.

보안 기술 문제는 접속이 용이하고, 해킹에 의한 정보 접근이 용이한 인터넷의 개방 지향적인 특성 때문에 완벽한 보안 시스템을 구축한다는 것은 그리 쉬운 문제가 아니다. 그러나, 허가된 사용자에 대해서는 최대한 거래 정보나 거래 내용 등과 같은 정보보호를 보장해 주는 시스템 구축이 중요하다 하겠다.

이러한 보안상의 문제점을 해결하기 위한 대안으로 제시되고 있는 것이 암호 알고리즘을 통한 시스템 구축이다. 누구나가 접근이 용이하도록 인터넷을 기반으로 하는 전자상거래에서 이러한 암호 알고리즘은 필수적인 선결 과제이다. 그러므로, 전자상거래를 실현하기 위한 핵심 요건 중에 하나가 안전성과 효율성을 고루 갖춘 암호 알고리즘의 구축이라 할 수 있다.

본 논문에서 제안한 NC(Nonpolynomial Complete) 암호 알고리즘은 암호화 기법의 분류 중 메시지를 블록 단위로 분할하여 암호화하는 블록 암호 알고리즘으로서 암·복호화 키를 동일하게 각각 128비트로 사용하였다.

NC 암호 알고리즘은 기본적으로 비선형 변환과 선형 변환의 적절한 조합에 의해 설계됐으며, 알고리즘의 전체 구조는 데이타 블록의 좌·우 측면에 교대로 비선형 변환을 적용시키는 페이스텔(Feistel) 구조를 적

용하여 설계하였다. 이를 바탕으로 암호문의 생성 속도가 빠르게 진행되며, 외부 공격에도 대처할 수 있도록 설계하였다[11].

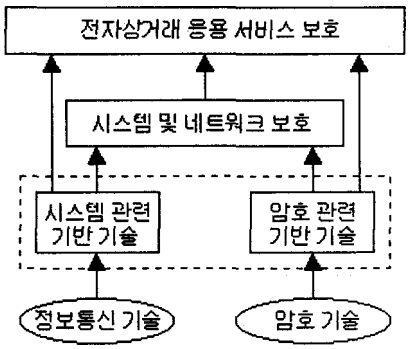
2. 전자상거래 상에서의 보안

전자상거래라 함은 통합적으로 자동화된 정보체계의 환경 하에서 기업과 기업간, 기업과 정부간, 기업과 개인간 거래관계의 모든 측면에 걸쳐 생산, 구매, 대금 지불, 수송, 행정, 서비스 등의 제반 비즈니스를 네트워크를 통해 전자적으로 행하는 것으로 정의할 수 있다. 인터넷에서 안전한 전자상거래를 보장하기 위해서는 거래 상대를 확인함과 동시에 거래 정보의 복제에 의한 정보의 부당한 누출을 방지하는 대책이 절실하다. 즉, 거래 상대에 대한 인증을 수행하고, 거래 내용에 부정이 없다는 것을 증명하는 구조가 갖춰져 있어야 하며, 거래 내용이 제 3 자에게 노출되지 않게 기밀성을 보장하는 암호화와 다른 사람에 의한 변조 또는 본인이 작성한 사실이나 내용을 부정할 수 없게 하는 전자서명 기능을 채택한 프로토콜 등이 필요하다[3]. 이는 정형화된 문서를 주고받는 전자문서 교환(EDI), 전자우편(E-mail), 팩스(FAX), 바코드(Barcode), 전자대금 이체(EFT), 전자정보 서비스, 음성사서함, 이미지 시스템, 비디오 메시징 등 그 종류와 형태가 매우 다양하다.

최근 네트워크의 발달에 따른 개방형 통신망에 있어서 각 개인이나 기업 또는 정부의 정보들이 인터넷을 통해 손쉽게 상대방에게 전달된다. 반대로 말하면, 이러한 정보들을 다른 사람이 인터넷을 통하여 손쉽게 접근 할 수 있다는 것을 의미한다. 더욱이 웹 기술의 발달에 따라 일반 사용자들도 이러한 정보들에 손쉽게 접근이 가능해짐으로서 인증이나 개인의 사생활 및 개인정보 등의 전자상거래 보안 문제는 더욱 더 중요한 과제로 대두되고 있다.

따라서, 전자상거래에서도 인터넷 보안 문제를 반드시 고려하여야 한다. 일반적인 인터넷 보안의 문제점을 살펴보면, UNIX 및 TCP/IP(Transmission Control Protocol/Internet Protocol) 프로토콜의 소스(source) 공개, 인터넷 서비스업체 및 상용 네트워크의 증가로 인한 손쉬운 접속, 해킹 방법에 관한 정보 접근의 용이성, 인터넷의 개방지향적 특성, 각종 응용 프로그램들에 존재하는 버그, 그리고 시스템 및 네트워크 관리자의 보안 의식 결여 등을 지적할 수 있다.

그러므로, 이러한 인터넷 상의 정보보호 문제를 포함한 보안 문제를 해결하기 위한 제반 장치가 마련되지 않는다면, 전자상거래의 안전성을 기대할 수가 없다. 전자상거래의 정보보호를 위한 기술은 <그림 1>과 같이 분류할 수 있다[2].



<그림 1> 전자상거래 정보보호 계층

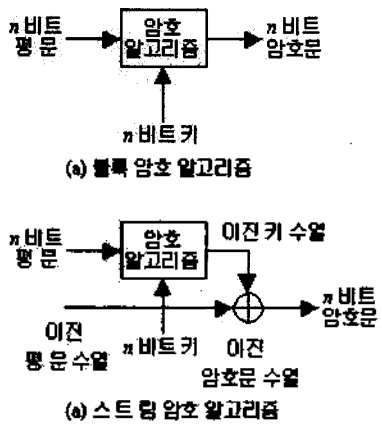
이러한 정보보호 문제를 해결하기 위한 근본적인 방법 중에 하나가 네트워크 상에서 송·수신 메시지나 거래 내용에 관한 정보를 암·복호화하여 사용하는 암호(또는 보안) 기술이다.

암호 기술은 암호키의 운용에 따라 대칭키 방식과 공개키 방식으로 대별된다. 대칭키 암호 시스템(대칭형)에서는 송·수신자가 동일한 비밀키를 공유해야 한다. 대규모의 개방형 통신망에서는 각 사용자가 상대방과 각각의 비밀키를 공유하는 것이 가능하므로, 통상 키 분배는 공개키 암호 시스템을 이용해서 이루어진다. 공개키 암호 시스템은(비대칭형) 암·복호화 키가 서로 다르며, 어느 한 키를 공개하더라도 대응되는 다른 키를 유도해 내는 것은 계산상 불가능하도록 설계되어진다.

암호키의 사용에서 알 수 있듯이 대칭키 방식은 두 사용자간에 비밀 채널이 형성되어 쌍방간의 실체 인증이나 메시지 인증 기능을 제공한다. 그러나, 공개키 방식에서는 한 방향, 즉, 공개키로 암호화하는 송신자로부터 해당 비밀키의 소유자에게로의 비밀 채널만이 형성되며 인증 기능은 없다. 이런 암호 기술은 암호화를 통해 적용된다.

암호화란 원래의 내용을 숨겨 원본과 달라진 전달문, 혹은 비밀문을 만드는 일련의 과정으로서, 한 문자를 다른 것으로 대체하는 일련의 규칙들의 집합이다. 즉, 이것은 컴퓨터 상에서 배타적 OR(exculsive OR: XOR)나 기타 다른 수학적인 기법들(알고리즘)을 통해 메시지와 결합된 문자의 조합을 만드는 것을 의미한다[6].

대칭키 암호 알고리즘은 메시지 처리 형식에 따라 다시 블록 암호 알고리즘과 스트림 암호 알고리즘으로 나누어지고, 공개키 암호 알고리즘은 키 분배 문제(또는 키 공유 문제)에 근거하여 공개키 분배 알고리즘과 공개키 관리 알고리즘으로 나누어질 수 있다.



<그림 2> 대칭키 암호 알고리즘의 형태

블록 암호 알고리즘은 고정된 크기의 입력 블록과 임의의 키에 의해 고정된 크기의 출력 블록으로 변형되는 암호 알고리즘으로서, 출력 비트의 각 비트는 입력 블록과 키의 모든 비트에 영향을 받아 결정된다. 이에 비해, 스트림 암호 알고리즘은 선형 쉬프트 레지스터를 이용한 이진 수열 발생기를 사용하는 암호 알고리즘의 형태로서, 평문을 이진 수열로 부호화 한 뒤, 이를 이진 수열 발생기에서 생성된 이진 수열과 XOR하여 이진 수열로 된 암호문을 산출하는 방식이다.

또한, 최근 들어 암호 알고리즘에 대한 관심과 사용이 증가하면서 키 위탁/복구 시스템(key recovery/escrow)에 대한 이슈가

대두되고 있다. 키 위탁 시스템은 특정 조건이 만족되었을 경우 평문을 쉽게 얻을 수 있도록 제 3 자에게 키를 위임하는 것이며, 키 복구 시스템은 특정 상황 하에서 암호문에 접근할 수 없을 때 사전에 약속된 허가받은 개체에게 복구 능력을 제공하는 것으로 정의하고 있다. 보통 키 복구 시스템은 논리적으로 사용자 보안 요소, 복구 기관 요소, 데이터 복구 요소의 세 가지 주요 구성요소로 구분할 수 있으며, 이들 구성요소의 성격에 따라 복구 시스템의 방식이 결정된다.

3. NC 블록 암호 알고리즘

3.1 NC의 소개

본 논문에서 제안한 NC 암호 알고리즘은 대칭키 암호 알고리즘에 바탕을 둔 개념으로서, 블록 단위로 메시지를 처리하는 블록 암호 알고리즘이다. 현대의 암호학에 있어서, 대칭키 암호 알고리즘은 메시지의 비밀성을 제공하는 암호 시스템의 중요한 요소이다. 보통 n 비트 블록 암호 알고리즘이란 고정된 n 비트 평문이 동일한 길이를 갖는 n 비트 암호문으로 바뀌는 것을 의미하는데, 여기서 n 비트는 블록의 크기를 나타낸다. 결국 이러한 변형 과정에서 동일한 키인 암호키와 복호키가 작용하여 암·복호화를 수행하게 되는 것이다[7].

일반적인 Feistel의 구조를 바탕으로 한 NC 암호 알고리즘은 입·출력문의 크기와 암호화에 사용되는 입력키의 크기가 각각 128비트이다. 또한, 효율성 측면에서 살펴볼 때 암·복호화 처리 시간이 7.63 Mbps인 NC가 15.10 Mbps 정도인 이전의 3-DES (triple-Data Encryption Standard)보다 빠르게 처리되며, 내부 함수 F에 관해서는 키 생성 알고리즘에 의해 생성되는 서브키와 비선형 함수 S-Box(Substitution Box)를 적용하여 설계했다. 또한, 키 생성 알고리즘은 라운드 동작과 동시에 암·복호화 라운드 키가 생성될 수 있도록 설계하였다. 이러한 설계 요구조건들을 바탕으로 제안한 NC 블록 암호 알고리즘은 다음과 같은 설계 기준들에 맞게 설계되었다.

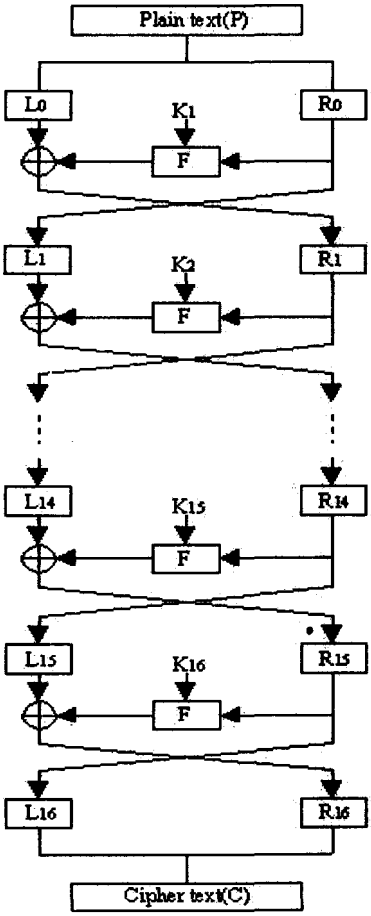
- 성격 : 128비트 블록 암호 알고리즘
- 구조 : Feistel 구조
- 입·출력문의 크기 : 128비트
- 입력키의 크기 : 128비트
- 서브키의 크기 : 64비트
- 라운드 수 : 16 라운드
- 내부 함수 F : 2개의 S-Box S_1, S_2와 64비트 서브키

3.2 NC의 설계

3.2.1 전체 구성

NC 알고리즘의 전체 구조는 대부분의 블록 암호 알고리즘에서 사용되고 있는 전형적인 Feistel 구조로 이루어져 있다[5].

다음의 <그림 3>은 전형적인 Feistel 구조를 바탕으로 본 논문에서 제안한 NC 블록 암호 알고리즘의 전체 구조를 도식화 한 것이다.



<그림 3> NC의 전체 구조

이런 구조는 메시지 처리 방식에 있어 128비트의 평문 블록 단위 당 128비트 암호키로부터 치환되어 생성된 64비트 서브키(16개)를 입력으로 받아 16 라운드를 수행한 후 128비트의 암호문 블록을 출력한다. 아울러, NC 암호 알고리즘은 전자상거래 상에서의 안전성과 효율성을 고려하여 설계하였다.

이것은 전자상거래 상에서 활용되는 전자 데이터의 기밀성 기능과 안전성을 제공하기 위해 평문을 암호문으로 바꾸는 암호 알고리즘으로서, 입력 평문 P의 키의 크기가 128비트인 하나의 블록을 동일한 크기의 키로 암·복호화 하기 위해 초기 치환 IP 테이블을 적용하여 128비트로 치환한다.

치환을 위한 IP 테이블은 128비트 평문을 16×8의 블록으로 분할하여 먼저 짝수 열의 순서로 홀수 행부터 역순으로 치환하고 이어서 짝수 행부터 다시 역순으로 치환한다. 그런 후에, 홀수 열의 순서로 홀수 행부터 역순으로 치환하고 이어서 짝수 행부터 다시 역순으로 치환하여 생성하였다. 예를 들어, <표 1>의 IP 테이블에서 114는 128비트 평문의 114번째 비트 위치의 값을 128비트 블록의 첫 번째 위치로 치환하고, 98은 98번째 비트 위치의 값을 두 번째 위치로 치환함을 의미한다. 최종적으로 15는 평문의 15번째 비트 위치의 값을 128비트 블록의 128번째 위치로 치환함을 의미한다.

<표 1> IP 데이블

114	98	82	66	50	34	18	2
116	100	84	68	52	36	20	4
118	102	86	70	54	38	22	6
120	104	88	72	56	40	24	8
122	106	90	74	58	42	26	10
124	108	92	76	60	44	28	12
126	110	94	78	62	46	30	14
128	112	96	80	64	48	32	16
113	97	81	65	49	33	17	1
115	99	83	67	51	35	19	3
117	101	85	69	53	37	21	5
119	103	87	71	55	39	23	7
121	105	89	73	57	41	25	9
123	107	91	75	59	43	27	11
125	109	93	77	61	45	29	13
127	111	95	79	63	47	31	15

그리고 나서, IP 테이블을 통해 치환된 128비트 블록을 각각 두 개의 64비트 블록 $L_0(64)$, $R_0(64)$로 분할하고, $L_i \oplus R_i$와 i번째 서브키를 나타내기 위해 K_i로 표현되는 16개의 64비트 서브키 $K_1(64)$, . . . , $K_{16}(64)$가 내부 함수 F 내에서의 조합에 의해 생성된 키 비트와 좌측키 L_i를 XOR 함으로서 첫 번째 라운드의 출력 값을 생성한다. 이 출력 값이 다음 라운드의 입력으로 설정되며, 이와 같은 수행 과정으로 16 라운드를 수행한 후, 최종적으로 결과 값을 산출하기 위해 역 치환 IP^{-1} 테이블로 치환된 128비트의 블록 $L_{16}(64)$, $R_{16}(64)$을 암호문 C로 출력하는 구조를 갖고 있다.

역 치환 IP^{-1} 테이블은 128비트 평문을 16×8의 블록으로 분할하여 9, 10행을 중심으로 8열부터 역순으로 시작 위치에서 위 열로 8비트 아래 열로 2비트씩 이동시켜 각각의 비트 위치 값을 치환하여 생성하였다.

<표 2> IP^{-1} 테이블

80	16	96	32	112	48	128	64
79	15	95	31	111	47	127	63
78	14	94	30	110	46	126	62
77	13	93	29	109	45	125	61
76	12	92	28	108	44	124	60
75	11	91	27	107	43	123	59
74	10	90	26	106	42	122	58
73	9	89	25	105	41	121	57
72	8	88	24	104	40	120	56
71	7	87	23	103	39	119	55
70	6	86	22	102	38	118	54
69	5	85	21	101	37	117	53
68	4	84	20	100	36	116	52
67	3	83	19	99	35	115	51
66	2	82	18	98	34	114	50
65	1	81	17	97	33	113	49

IP 테이블과 IP^{-1} 테이블은 각각 입력 평문을 치환하고 그것을 암호문으로 최종 생성하기 이전에 역으로 치환하는 테이블로서 이 두 테이블은 차분 공격시 역으로 암호화에 사용된 키를 찾는데 있어 몇 단계를 더 거침으로서 외부 공격에 의한 계산상의 복잡성을 증가시키기 위한 테이블들이다.

위에서 i번째 반복 라운드의 결과를 C_i라고 하고, L_i와 R_i를 각각 P_i의 좌측 평문과 우측 평문이라 하자. 그러면, $C_i = L_iR_i$로 표현할 수 있는데, 여기서 좌측 평문 L_i는 $L_i = t_1t_2 \cdot \cdot \cdot t_{63}t_{64}$, 우측 평문 R_i는 $R_i = t_{65}t_{66} \cdot \cdot \cdot t_{127}t_{128}$ 이고 $t_k(1 \le k \le 128)$는 L_i와 R_i의 k번째 비트를 표기한다.

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

여기서, K_i는 키 생성 알고리즘에 의해 생성되는 64비트 서브키로서, 보다 세부적인 과정은 다음 절에서 설명한다. 주의할 것은 최종 라운드 이후에는 좌·우측이 교체되지 않는다는 것이다. 그러나, 프로그램상에서는 알고리즘의 복잡성 분석을 위해 "Configure" 탭에서 좌·우측 결과 값 교체도 가능하도록 구현하였는데, 이것은 최종적으로 좌·우측 결과 값이 교체된 상태에서의 복호화 과정에서는 다시 좌·우측 결과 값을 교체한 후 역으로 복호화를 수행해야하므로 그 과정을 보다 복잡하게 구성할 수 있기 때문이다. 다시 말해서, 입력으로 들어온 우측 평문 키가 내부 함수 F 내에서의 조합을 거친 이후에 좌측 평문 키와 XOR되어 결과를 출력한 이후에 그 키 비

트가 상호 교체되지 않고 그대로 출력된다는 것이다. 결과적으로 블록 $L_{16}R_{16}$이 최종 수열로의 입력이 되어 암호문을 생성한다. 이것은 암·복호화를 사용하는 알고리즘에 있어 필요하다.

3.2.2 키 생성 알고리즘

내부 함수 F에서 사용되는 서브키를 생성하기 위한 키 생성 알고리즘의 가장 중요한 설계 목표는 안전성을 보장하는 것이다. 그러므로, 128비트 암호 알고리즘에 있어서는 128비트 키를 이용하여 64비트 서브키를 산출하게 된다. 최근에는 차세대 표준인 AES(Advanced Encryption Standard) 암호 알고리즘의 영향으로 128비트 블록 암호가 선호되고 있는데[13], 기존의 Feistel 구조로 설계한다면 64비트 입·출력의 서브키가 필요하다. 그런데, 64비트 입·출력의 내부 함수는 DES(Data Encryption Standard) 암호 알고리즘과 같은 기존의 32비트 입·출력의 내부 함수에 비하여 구성이 어려우며, 차분 특성 확률의 상한을 계산하는 것도 어려운 일이다. 따라서, Feistel 구조와 같이 입력을 두 개로 나누는 방법을 일반화시켜 MARS, RC6, Twofish, Skipjack 알고리즘 등과 같이 블록을 4개로 나누어 한 라운드를 구성하는 방법이 제안되고 있다.

각 라운드에서 내부 함수 F 내부의 수열과 밀접한 연관이 있는 2^{128}의 하나에 따라 선택되는 라운드 서브키를 생성하기 위한 키 생성 알고리즘은 128비트 평문 블록을 두 개의 64비트 블록으로 분할하고, 이를 다시 좌·우측 각 2개의 32비트 키 값(A, B, C, D)를 이용하여 서브키를 생성한다. 이를 위해 임의의 128비트 키를 128비트 키로 재배열하기 위해 PC1 테이블을 이용한다.

PC1 테이블은 128비트 평문을 16×8의 블록으로 분할하여 1, 2행은 1열의 16행부터 역으로 치환하고, 3, 4행은 2열의 9, 16,…10행, 1, 8…2행으로, 5, 6행은 3열의 10, 9, 16…11행, 2, 1, 8…3행으로, 7, 8행은 4열의 11, 10, 9, 16…12행, 3, 2, 1, 8…4행으로 각각 치환하고, 9, 10행은 8열의 16행부터 역으로 치환하고, 11, 12행은 7열의 9, 16,…10행, 1, 8…2행으로, 13, 14행은 6열의 10, 9, 16…11행, 2, 1, 8…3행으로, 15, 16행은 5열의 11, 10, 9, 16…12행, 3, 2, 1, 9…4행으로 각각 치환하여 생성하였다.

<표 3> PC1 테이블

121	113	105	97	89	81	73	65
57	49	41	33	25	17	9	1
66	122	114	106	98	90	82	74
2	58	50	42	34	26	18	10
75	67	123	115	107	99	91	83
11	3	59	51	43	35	27	19
84	76	68	124	116	108	100	92
20	12	4	60	52	44	36	28
128	120	112	104	96	88	80	72
64	56	48	40	32	24	16	8
71	127	119	111	103	95	87	79
7	63	55	47	39	31	23	15
78	70	126	118	110	102	94	86
14	6	62	54	46	38	30	22
85	77	69	125	117	109	101	93
21	13	5	61	53	45	37	29

이 때, n 라운드에서의 서브키는 n-1 라운드에서 생성된 서브키에 영향을 받도록 설계했다. 이를 위해 n-1 라운드에서 생성된 서브키를 Shift 테이블에 적용하여 규칙

적으로 1 또는 3비트씩 각 라운드별로 좌측 회전이동 한 후, n 라운드의 서브키로 적용하였는데, 이것은 64비트 서브키의 절반에 해당되는 32비트의 위치를 Shift 시킴으로서 16 라운드 후에는 서브키의 좌·우측 블록의 키 위치가 바뀌게 된다. 또한, NC에서는 키 생성 알고리즘의 구조상 서브키를 생성하기 위해서는 Shift를 수행한 후에 연산이 이루어지므로 일정하게 Shift를 하게 되면 어떤 특정 비트 위치에 대해서만 계산이 가중되게 되므로 이를 방지하기 위해서 규칙적인 Shift 테이블을 적용하였다.

<표 4> Shift 테이블

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

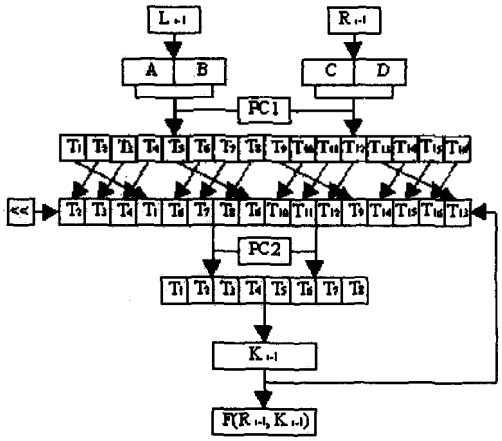
그런 후, 128비트 키를 64비트로 치환하여 산출하기 위해 PC2 테이블을 이용한다.

PC2 테이블은 Shift 테이블을 통해 회전이동된 128비트 평문을 16×8의 블록으로 분할하여 행의 4비트 전·후 위치 비트 값과 열의 2배수 위치 비트 값을 상호 역순으로 교차하여 치환하도록 생성하였다.

<표 5> PC2 테이블

31	1	23	9	63	33	55	41
29	3	21	11	61	35	53	43
27	5	19	13	59	37	51	45
25	7	17	15	57	39	49	47
95	65	87	73	127	97	119	105
93	67	85	75	125	99	117	107
91	69	83	77	123	101	115	109
89	71	81	79	121	103	113	111

<그림 4>는 NC 암호 알고리즘에 대한 키 생성 알고리즘의 구조를 나타낸 것이다.



<그림 4> 키 생성 알고리즘의 구조

평문의 우측 키 값과 함께 내부 함수 F의 입력 키 값으로 사용되는 64비트 서브키 생성을 위한 이런 키 생성 알고리즘 구조는 기존의 Feistel 구조와는 달리 내부 함수 F의 출력 결과 값이 다음 라운드에 영향을 미치면서 동시에 입력 블록을 변화시키기 때문에 데이터의 복잡도를 빠르게 증가시킬 수 있고, 또한 암·복호화가 빠르게 처리된다는 관점에서 좋은 키 스케줄링 구조라 할 수 있다.

즉, 64비트 키는 덧셈 모듈 2^{64}을 이용한 각 라운드의 내부 함수 F의 출력으로 조합된다. 또한, 라운드 서브키들은 외부 공격을 방지하기 위해 매 라운드마다 다소 달라야 한다. 이러한 라운드 서브키는 S-Box의 병합과 함께 서브키 생성이 보다 복잡하도록 내부 함수 F를 구축하지만, 정확하게 어떻게 암호 알고리즘 작업을 예측할 수 있는지에 있어 보다 유용하게 사용된다.

그리고, 키 생성 알고리즘은 기본적으로 모든 라운드 서브키를 저장할 수 없는 하드웨어적 한계나 제한된 자원을 갖는 스마트카드와 같은 응용에서의 효율성을 위하여 암호화나 복호화시 암호키로부터 필요한 라운드 키를 간단히 적용할 수 있도록 설계하였다.

NC 알고리즘의 내부 함수 F의 입력 연산을 위한 각 라운드에 사용되는 서브키는 다음과 같은 키 스케줄링 전개 과정을 통해 생성된다.

- 주어진 128비트 암호키 K를 선택한다.
- 이 암호키 K를 4개의 블록 A, B, C, D로 분할한다.
- K를 <표 3>에 적용하여 128비트로 치환한다(A_0, B_0, C_0, D_0).
  (PC1_Table[16*8] ← K)

<표 3>을 이용한 1차 서브키의 생성 과정은 다음과 같은 정리와 이를 이용한 조합 및 치환에 의해 전개된다.

- C_K : 128비트 임의의 키 값.
- T_i : C_K의 8비트 블록 분할 값 (i = 1, . . . ,16).
- a~p : T_i의 8비트 값.
- S_K : T_i의 조합 값(S_K = T_i[16]).

T_i가 $T_1 = p_1o_1m_1m_1l_1k_1j_1i_1$, $T_2 = h_1g_1f_1e_1 d_1c_1b_1a_1$, $T_3 = i_2p_2o_2n_2m_2l_2k_2j_2$, $T_4 = a_2h_2g_2f_2 e_2d_2c_2b_2$, $T_5 = j_3i_3p_3o_3n_3m_3l_3k_3$, $T_6 = b_3a_3h_3g_3 f_3e_3d_3c_3$, $T_7 = k_4j_4i_4p_4o_4n_4m_4l_4$, $T_8 = a_4b_4a_4h_4 g_4f_4e_4d_4$, $T_9 = p_8o_8n_8m_8l_8k_8j_8i_8$, $T_{10} = h_8g_8f_8e_8 d_8c_8b_8a_8$, $T_{11} = i_7p_7o_7n_7m_7l_7k_7j_7$, $T_{12} = a_7h_7g_7 f_7e_7d_7c_7b_7$, $T_{13} = j_6i_6p_6o_6n_6m_6l_6k_6$, $T_{14} = b_6a_6 h_6g_6f_6e_6d_6c_6$, $T_{15} = k_5j_5i_5p_5o_5n_5\ m_5l_5$, $T_{16} = c_5 b_5a_5h_5g_5f_5e_5d_5$로 전개되므로 여기서 S_K의 조합은 $S_K = T_1T_2T_3T_4T_5T_6T_7T_8T_9T_{10}T_{11}T_{12}T_{13} T_{14}T_{15}T_{16}$ 이다.

- A_0, B_0, C_0, D_0를 1비트 left shift 시킨 것을 각각 A_1, B_1, C_1, D_1 이라 한다. ($A_1B_1C_1D_1$ ← S_K <<= 1)
- A_1, B_1, C_1, D_1을 <표 5>에 적용하여 64비트로 치환한다(PC2_Table[8*8] ← K).

<표 5>를 이용한 2차 서브키의 생성 과정은 T_i가 $T_1 = d_7a_1c_7b_1h_7e_1g_7f_1$, $T_2 = d_5a_3c_5b_3h_5e_3g_5f_3$, $T_3 = d_3a_5c_3b_5h_3e_5g_3f_5$, $T_4 = d_1a_7c_1b_7h_1e_7g_1f_7$, $T_5 = l_7i_1k_7j_1p_7m_1o_7n_1$, $T_6 = l_5i_3k_5j_3p_5m_3o_5n_3$, $T_7 = l_3i_5k_3j_5p_3m_5o_3n_5$, $T_8 = l_1i_7k_1j_7p_1m_7o_1n_7$로 전개되므로 S_K는 $S_K = T_1T_2T_3T_4T_5T_6T_7T_8$ 이다.

- <표 5>에 의해 치환된 결과를 K_1 서브키라 한다(K_1 ← PC2).
- A_1, B_1, C_1, D_1을 1비트 left shift 시켜 각각 A_2, B_2, C_2, D_2를 산출한다. ($A_2B_2C_2D_2$ ← S_K <<= 1)
- A_2, B_2, C_2, D_2를 <표 5>에 적용하여 64비트로 치환한다(PC2_Table[8*8] ← K).
- <표 5>에 의해 치환된 결과를 K_2 서브키라 한다(K_2 ← PC2).
- A_2, B_2, C_2, D_2를 3비트 left shift 시킨 것을 각각 A_3, B_3, C_3, D_3 라 한다. ($A_3B_3C_3D_3$ ← S_K <<= 3)
- A_3, B_3, C_3, D_3를 <표 5>에 적용하여 64비트로 치환한다(PC2_Table[8*8] ← K).

- <표 5>에 의해 치환된 결과를 K_3 서브키라 한다(K_3 ← PC2).
- 같은 방법으로 계속 수행하여 K_{16} 서브키를 산출한다.

3.2.3 S-Box

대부분의 블록 암호 알고리즘에서 서브키의 입·출력과 관련하여 내부 함수 F 내에서 비선형 대입 연산을 수행하는데 사용되는 S-Box(Substitution Box)는 비선형 함수로서 입력 크기와 출력 크기를 변경할 수 있으며, 임의적으로 생성시킬 수 있다.

S-Box는 1977년 Lucifer 알고리즘에서 최초로 사용되었으며, 그 후 대개의 암호 알고리즘에서 널리 사용되고 있다[10]. 이러한 S-Box는 암호화에 민감한 영향을 미치므로 S-Box의 구성을 어떻게 하느냐에 따라 견고한 암호 알고리즘을 구축할 수 있다. 결국, 암호 알고리즘의 핵심 사항은 견고한 S-Box의 새로운 설계나 또는 실험을 통해 검증된 S-Box를 적용하는 것이다.

제안한 NC 알고리즘에서는 2개의 8×8 비트 S-Box S_1과 S_2를 사용하였으며, 여기서 각각의 S-Box는 NC 알고리즘의 키 크기에 의존한다. 즉, 암·복호화 과정을 어렵게 하기 위해 각 라운드간에 키-의존 관계를 갖도록 설계되어 입증된 2개의 S-Box를 사용하였다[13].

전형적으로 S-Box의 형태는 비선형이라고 알려져 있다. 즉, 입력 쌍의 XOR 내용은 출력 쌍의 XOR 내용을 보장할 수 없으므로, 일반적으로 여러 개의 출력 XOR가 가능하다. 이것은 양쪽 입력이 같을 때 특별한 경우가 발생한다. 그러나, 중요한 관점은 각각의 입력 XOR에 대해 모든 출력 XOR가 가능한 것은 아니며, 가능한 어느 하나도 일정하게 나타나지 않으며, 몇몇 XOR 값은 다른 것보다 더욱 빈번하게 나타난다. 이런 S-Box의 특성은 암호 알고리즘의 안전성 분석에 유용하게 적용된다.

NC 알고리즘에서는 각각의 8×8 비트 S-Box를 16진수 표기법을 이용한 개체들의 목록으로 표현했다. 다음의 <표 6>과 <표 7>은 각각 S-Box S_1과 S_2의 목록을 나타낸 것이다.

<표 6> S_1 테이블

S_1	S_0	S_1	S_0	S_1	S_0	S_1	S_0	S_1	S_0	S_1	S_0	S_1	S_0
00	A9	01	85	02	D6	03	D3	04	54	05	1D	06	AC	07	25
08	5D	09	43	0A	18	0B	1E	0C	51	0D	FC	0E	CA	0F	63
10	28	11	44	12	20	13	9D	14	E0	15	E2	16	C8	17	17
18	A5	19	8F	1A	03	1B	79	1C	BA	1D	13	1E	D2	1F	EE
20	70	21	92	22	3F	23	A8	24	32	25	DC	26	F6	27	74
28	EC	29	95	2A	0B	2B	57	2C	5C	2D	5B	2E	B3	2F	01
30	25	31	1C	32	73	33	98	34	10	35	CC	36	F2	37	D9
38	2C	39	E7	3A	72	3B	83	3C	9B	3D	B1	3E	86	3F	C9
40	60	41	50	42	A3	43	EB	44	0D	45	B6	46	9E	47	4F
48	B7	49	5A	4A	C6	4B	78	4C	A6	4D	12	4E	AE	4F	D5
50	61	51	C3	52	B4	53	41	54	52	55	7D	56	8D	57	08
58	1F	59	99	5A	00	5B	19	5C	04	5D	53	5E	F7	5F	D1
60	FD	61	76	62	2F	63	27	64	B0	65	8B	66	0E	67	AE
68	A2	69	6E	6A	93	6B	4D	6C	09	6D	7C	6E	09	6F	0A
70	BF	71	EF	72	F3	73	C5	74	87	75	14	76	FE	77	64
78	DE	79	2E	7A	4B	7B	1A	7C	06	7D	21	7E	6B	7F	66
80	02	81	F5	82	92	83	8A	84	0C	85	B3	86	7E	87	D0
88	7A	89	47	8A	96	8B	E5	8C	26	8D	80	8E	AL	8F	DF
90	A1	91	30	92	37	93	AE	94	36	95	15	96	22	97	38
98	F4	99	A7	9A	45	9B	4C	9C	81	9D	E9	9E	84	9F	97
A0	35	A1	CB	A2	CE	A3	3C	A4	71	A5	11	A6	C7	A7	89
A8	75	A9	FB	AA	DA	AB	F8	AC	A4	AL	59	AE	02	AF	C4
B0	FF	B1	49	B2	39	B3	67	B4	C0	B5	CF	B6	D7	B7	B8
B8	0F	B9	8E	BA	42	BB	23	BC	91	BD	66	BE	DB	BF	A4
C0	34	C1	F1	C2	48	C3	C2	C4	6F	C5	3D	C6	2D	C7	40
C8	BE	C9	3C	CA	BC	CB	C1	CC	AA	CD	BA	CE	4E	CF	55
D0	3B	D1	DC	D2	68	D3	7F	D4	9C	D5	D8	D6	4A	D7	56
D8	77	D9	A0	DA	ED	DB	46	DC	B5	DL	2B	DE	65	DF	FA
E0	E3	E1	B9	E2	B1	E3	9F	E4	5E	E5	F9	E6	E6	E7	B2
E8	31	E9	EA	EA	6D	EB	5F	EC	E4	ED	F0	EE	CD	EF	88
F0	16	F1	3A	F2	58	F3	D4	F4	62	F5	29	F6	07	F7	33
F8	E6	F9	1B	FA	05	FB	79	FC	90	FD	6A	FE	2A	FF	9A

<표 7> S_2 테이블

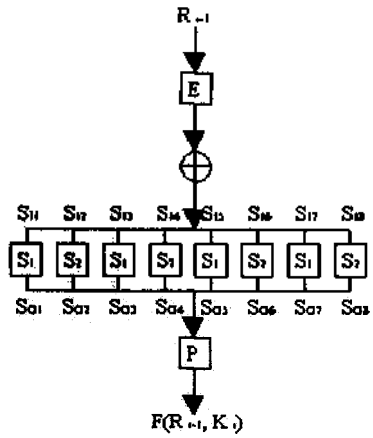
S_I	S_O	S_I	S_O	S_I	S_O	S_I	S_O	S_I	S_O	S_I	S_O	S_I	S_O	S_I	S_O
00	38	01	E8	02	2D	03	A6	04	CF	05	DE	06	B3	07	B8
08	AF	09	60	0A	55	0B	C7	0C	44	0D	6F	0E	6B	0F	5B
10	C3	11	62	12	33	13	B5	14	29	15	A0	16	E2	17	A7
18	D3	19	91	1A	11	1B	06	1C	1C	1D	BC	1E	36	1F	2B
20	EF	21	88	22	6C	23	A8	24	17	25	C4	26	16	27	F4
28	C2	29	45	2A	E1	2B	D6	2C	3F	2D	3D	2E	92	2F	9C
30	28	31	4E	32	F6	33	3E	34	A5	35	F9	36	0D	37	DF
38	D8	39	2B	3A	66	3B	7A	3C	27	3D	4F	3E	F1	3F	72
40	42	41	D4	42	41	43	C0	44	73	45	67	46	AC	47	8B
48	F7	49	AD	4A	80	4B	1F	4C	CA	4D	2C	4E	AA	4F	34
50	D2	51	0B	52	EE	53	E9	54	5D	55	94	56	18	57	F8
58	57	59	AE	5A	08	5B	C5	5C	13	5D	CD	5E	86	5F	B9
60	FF	61	7E	62	C1	63	31	64	F5	65	0A	66	6A	67	B1
68	D1	69	20	6A	D7	6B	02	6C	22	6D	04	6E	68	6F	71
70	07	71	DB	72	9D	73	99	74	61	75	BE	76	E6	77	59
78	DC	79	51	7A	90	7B	DC	7C	9A	7D	A4	7E	AE	7F	D0
80	81	81	0F	82	47	83	1A	84	E3	85	EC	86	8D	87	BF
88	96	89	7B	8A	5C	8B	A2	8C	A1	8D	63	8E	23	8F	4D
90	C8	91	9E	92	98	93	3A	94	0C	95	2E	96	BA	97	6E
98	9F	99	5A	9A	F2	9B	92	9C	F3	9D	49	9E	78	9F	3C
A0	15	A1	FB	A2	70	A3	75	A4	7F	A5	35	A6	10	A7	03
A8	64	A9	6D	AA	C6	AB	74	AC	D5	AD	B4	AE	EA	AF	09
B0	76	B1	19	B2	FE	B3	40	B4	12	B5	E0	B6	BD	B7	05
B8	FA	B9	01	BA	F0	BB	2A	BC	5E	BD	A9	BE	56	BF	43
C0	85	C1	14	C2	89	C3	9B	C4	B0	C5	E5	C6	48	C7	79
C8	97	C9	FC	CA	1E	CB	82	CC	21	CD	8C	CE	1B	CF	5F
D0	77	D1	54	D2	B2	D3	1C	D4	15	D5	1F	D6	00	D7	46
D8	ED	D9	58	DA	52	DB	EB	DC	7E	DD	DA	DE	C9	DF	FD
E0	30	E1	95	E2	65	E3	3C	E4	B6	E5	E4	E6	BB	E7	7C
E8	0E	E9	50	EA	39	EB	26	EC	32	ED	84	EE	69	EF	93
F0	37	F1	E7	F2	24	F3	A4	F4	CB	F5	53	F6	0A	F7	87
F8	D9	F9	4C	FA	83	FB	8F	FC	CE	FD	3B	FE	4A	FF	B7

예를 들어, S-Box S_1의 목록에서 S_I = 00일 때 S_O = 10101001을 16진수 A9로, S_I = 01일 때 S_O = 10000101을 16진수 85로 각각 표현한 것이다. 또한, S-Box S_2의 목록에서 S_I = 00일 때 S_O = 00111000을 16진수 38로, S_I = 01일 때 S_O = 11101000을 16진수 E8로 각각 표현한 것이다.

3.2.4 내부 함수 F

제안한 NC 암호 알고리즘에서 128비트 키를 통한 암·복호화를 위한 근본적인 블록 구축은 평문의 우측 비트 값 64비트와 키 생성 알고리즘에 의해 생성되는 각 라운드의 서브키 64비트를 입력 비트 값으로 하여 그것들의 비트 값을 XOR 함으로서 출력 비트 값을 산출하는 내부 함수 F 내에서 이루어진다.

다음의 <그림 5>는 NC 알고리즘 설계상에서의 내부 함수 F의 구조를 나타낸 것이다.



<그림 5> 내부 함수 F 구조

이런 내부 함수 F의 구조는 입력 문자열과 출력 문자열이 사상되는 키-의존 관계를 갖고 있다. 다시 말해서, 키 생성 알고리즘(PC1, PC2 테이블과 left shift 연산)에 의해 생성된 64비트의 서브키와 2개의 각 8비트 S-Box를 이용하여 데이터의 우측 면에서 수행된다. 내부 함수 F는 일반적으로 n개의 S-Box, 비트 수열, 수학적 연산 그리고 XOR에 근거하고 있기 때문에, 여기서는 2개의 S-Box만 이용했다. 또한, 내부 함수 F는 항상 비선형 형태를 갖는다.

$$F : \{0,1\}^{n/2} \times \{0,1\}^{N} \mapsto \{0,1\}^{n/2}$$

여기서 n은 NC 구조의 블록 크기이고

F는 입력처럼 블록의 $n/2$ 비트와 키의 N 비트를 취하고 $n/2$ 비트 길이의 출력을 산출하는 내부 함수이다. 각 라운드에서 원본 블록은 내부 함수 F로의 입력이고 내부 함수 F의 출력은 그것들의 2개 블록이 다음 라운드를 위해 교체된 후에 목표 블록과 XOR 된다. 그러므로, 전형적인 Feistel 구조를 갖는 NC 블록 암호 알고리즘은 내부 함수 F의 특성에 따라 입·출력을 구분할 수 있다.

<그림 5>를 바탕으로 내부 함수 F 내에서의 처리 과정을 보다 세부적으로 서술하면 다음과 같다.

- NC 알고리즘에서 내부 함수 F는 첫 번째로 우측 평문 값 64비트 블록을 입력으로 받아 외부 공격에 의한 암호 해독을 복잡하게 하기 위해 블록 내의 위치를 재정렬 하는 E 테이블을 통해 치환된 64비트와 PC1 테이블과 PC2 테이블에 의해 생성된 서브키 K_i를 XOR 한다.

$$E(R_{i-1})\oplus b_1b_2b_3b_4b_5b_6b_7b_8$$

<표 8> E 테이블

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

여기서, E 테이블은 64비트 서브키를 8×8의 블록으로 분할하여 각 행별로 홀수열의 내림차순과 짝수 열의 오름차순 조합으로 치환하여 생성하였다.

- XOR 이후에 산출된 각각의 8비트 블록 $B_i = b_1b_2b_3b_4b_5b_6b_7b_8$는 2개의 S-Box S_1과 S_2에 적용되어 $(S_1(B_1)S_2(B_2)S_1(B_3)S_2(B_4)S_1(B_5)S_2(B_6)S_1(B_7)S_2(B_8))$ 순서로 8 비트 블록 $S_i(B_i)$로 출력되고, 모든 블록들은 계산 복잡도를 증가시키기 위해 블록의 위치를 재정렬하는 P 테이블에 의해 치환된다.

$$P(S_1(B_1)S_2(B_2)S_1(B_3)S_2(B_4)\\ S_1(B_5)S_2(B_6)S_1(B_7)S_2(B_8))$$

<표 9> P 테이블

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

P 테이블은 64비트 서브키를 8×8의 블록으로 분할하여 각 블록별로 임의의 비트 위치에 E 테이블을 통해 치환된 64비트 서브키를 다시 치환하도록 설계하였다.

- 이 결과로 64비트 블록 $F(R_{i-1},K_i)$가 출력되며, 최종적으로 $L_{i-1}\oplus F(R_{i-1},K_i)$는 다음 라운드의 우측 입력 비트로 이동하고, 이런 과정을 i 라운드 만큼 수행한 후에 최종 암호문 쌍을 출력한다.

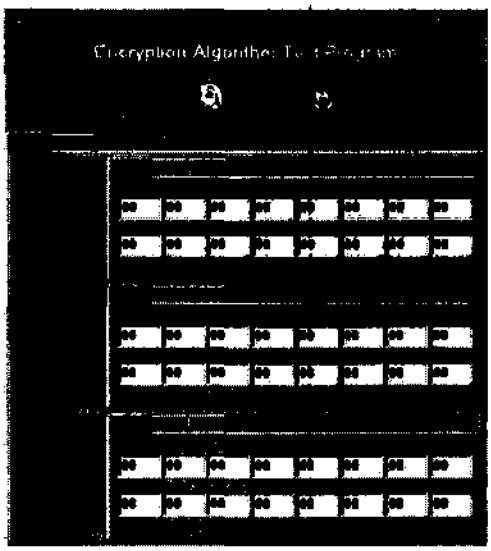
3.3 NC의 구현

3.3.1 구현 환경

NC 암호 알고리즘을 구현하기 위한 환경은 다음과 같다.

- 시스템 : Pentium Pro 200MHz 이상
- 메모리 : 64MB 이상
- OS : Windows 95/98
- 컴파일러 : Borland C++ Builder 4.0

3.3.2 초기화면 구성

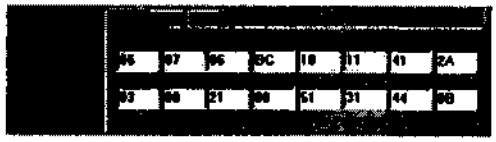


3.3.3 암·복호화 과정

NC 알고리즘에서 암호화 과정은 평문과 암호문의 크기가 각각 128비트이고 암호키의 크기 역시 128비트로 구성된다. 내부적으로 임의의 128비트 암호키는 키 생성 알고리즘에 의해 1번의 PC1 테이블 치환, 16번의 PC2 테이블 치환, 그리고 16번의 left shift 테이블에 의해 산출된 16개(K_1, . . . ,K_{16})의 서브키를 이용하여 16 라운드를 수행한다.

다음과 같이 각각 임의의 128비트 평문과 128비트 암호키를 입력하여 128비트 암호문을 생성하는 과정을 보자.

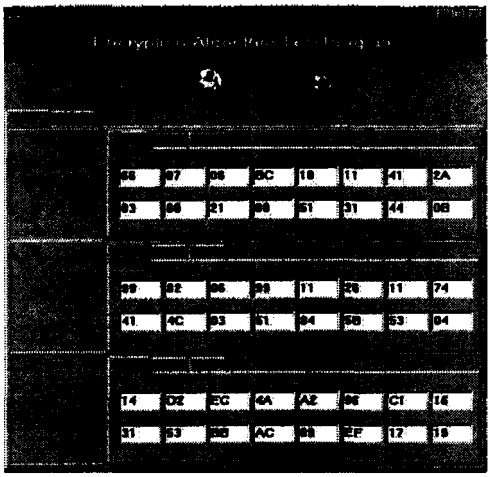
NC 프로그램 초기화면에서 "평문"을 클릭하여 다음과 같은 임의의 128비트 평문을 생성하고,



"암호키"를 클릭하여 다음과 같은 임의의 128비트 암호키를 생성한다. 생성된 128비트 암호키는 본 논문에서 제안한 키 생성 알고리즘의 과정을 거쳐 16개의 64비트 서브키를 산출한다.

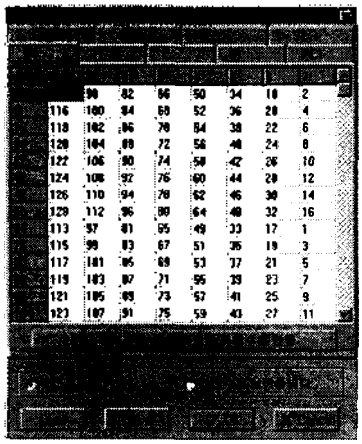
그런 후에, "Encoding" 버튼을 클릭하여 최종적으로 128비트 암호문을 산출한다.

또한, 구현한 프로그램에서는 암·복호화 과정의 정보에 관해 사용자 또는 관리자가 요구하는 제반 사항에 대한 몇 가지 분석 정보와 옵션들을 제공했다.

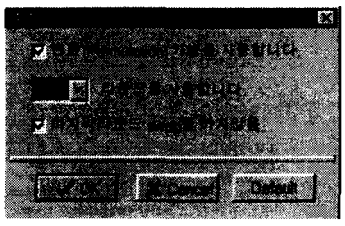


우선, 임의의 128비트 입력키에 대해 키 생성 알고리즘에 의해 생성된 16개 64비트 서브키의 목록에 관한 사항은 "DEBUG" 탭을 이용하여 확인할 수가 있다.

그리고, 암호화에 사용된 각종 파라메타들에 관한 내용을 참고하고 싶을 경우에는 "Parameter" 버튼을 클릭하여 각 테이블에 관한 사항을 확인한다.



또한, 수행되는 라운드 수의 조정과 변환 기능, 그리고 설정된 최종 라운드에서의 swap 여부 등을 컨트롤할 수 있게 구현하였다. 이것은 프로그램에 의해 산출된 128비트의 암호문을 해독하는 과정에 있어서 전수조사 공격(exhaustive attack)이나 차분공격(differential attack) 등으로 암호해독을 수행할 때, 수작업으로 3 라운드 정도는 3-라운드 암호해독 알고리즘을 통해 확인해 볼 수 있으므로, 초기 화면 구성에서 "변환(Permutation) 기능을 사용합니다."와 "마지막라운드 swap을 하지 않음"을 모두 해제하고 라운드 수를 3 라운드로 설정하여 수행하면 위의 공격법에 의한 산출된 결과와 비교해 볼 수 있다.



본 논문에서는 복호화 하는 과정에 대해서는 생략하는데, 복호화의 과정 역시 암호화의 과정에서 산출된 암호문과 동일한 암호키를 사용하여 "Decoding"을 클릭하면 평문을 산출할 수 있다.

3.4 비교 분석

본 논문에서는 각각의 블록 암호 알고리즘에서 사용된 키 생성 알고리즘에 근거한 서브키 생성 시간과 전체 암·복호화 처리 시간을 알고리즘 속도 측정 프로그램을 기반으로 블록 당 사이클(cycle per block) 속도로 비교 평가하였다.

이것은 동일한 아키텍처 하에서 클럭(clock)에 비례하는 속도를 구하기가 용이하기 때문에 구조가 서로 다른 알고리즘을 평가하는 경우에는 매우 적합하게 사용되는 측정 방법이다[14].

테스팅을 위한 실험 환경은 다음과 같이 설정하였다.

- 시스템 : Pentium II
- OS : Windows NT
- 컴파일러 : MS Visual C/C++ 5.0
- CPU : 200MHz
- 메모리 : 64MB

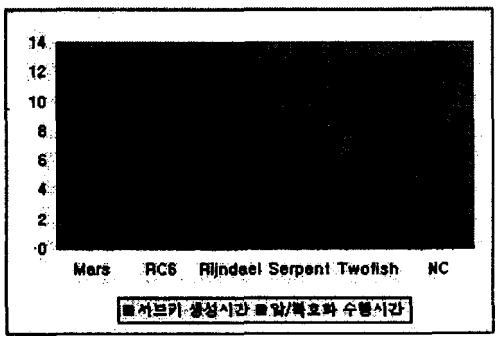
그리고, 실험 범위는 다음과 같이 설정하였다.

- 서브키 생성 시간은 임의의 키와 고정 키를 사용하는 암호키에 대해 서브키 생성 알고리즘을 반복적으로 수행하여 이를 평균으로 산출한다.
- 암/복호화 처리 시간은 임의의 단위 블록을 반복적으로 암호화하는 과정과 이것을 반복적으로 복호화하는 과정을 수행하여 이를 평균으로 산출한다.
- 각 알고리즘에 대해 서브키 생성 시간과 암/복호화 처리 시간을 측정하기 위해 동일하게 10초간 수행하여 그 속도를 측정한다.

다음의 <표 10>은 위의 실험 환경과 실험 범위를 바탕으로 NC 알고리즘과 기존의 암호 알고리즘들의 성능을 비교한 것이고, <그림 6>은 128비트 암호 알고리즘들의 성능을 그래프를 이용하여 비교한 것이다.

<표 10> NC 알고리즘과 기존 암호 알고리즘들과의 성능 비교

알고리즘명	서브키 생성 시간	암/복호화 처리 시간
Mars	10.11 μsec	8.27 Mbps
RC6	5.62 μsec	12.56 Mbps
Rijndael	7.97 μsec	10.78 Mbps
Serpent	8.03 μsec	8.21 Mbps
Twofish	4.66 μsec	11.82 Mbps
NC	2.42 μsec	7.63 Mbps



<그림 6> 128비트 암호 알고리즘의 비교

위의 실험 결과에서 NC 알고리즘의 서브키 생성 속도가 2.42 μsec 정도로 측정됐는데, 이 정도의 서브키 생성 속도라면 MAC(Messsage Authentication Code) 이나 해쉬 함수의 블록 구축시 상당히 효율적으로 사용될 수 있다.

4. 결 론

안전하게 전자상거래가 이루어지기 위해서는 이를 뒷받침해 줄 수 있는 장치가 필요한데, 그런 장치들 중에 하나가 바로 안전하고 효율적인 암호 알고리즘을 구축하는 것이라 할 수 있겠다.

본 논문에서 제안한 NC 암호 알고리즘은 이러한 암호 체계에 적합하도록 안전성과 효율성을 고려하여 설계했으며, 키의 크기를 128비트로 설계했으며, AES 블록 암호 알고리즘에 적합한 조건들을 최대한 수용하여 설계하였다.

또한, 일반적으로 암호화에 있어서 영향을 미치는 부분이 바로 암호키의 연산에 따라 이에 대응되는 키 비트 값이 결정되는 S-Box와 서브키를 생성하기 위한 키 생성 스케줄링인데, 이를 위해 검증된 두 개의 S-Box S_1과 S_2를 사용하였고, 외부 공격에 의해 암호키가 쉽게 발견되지 않도록 키 생성 알고리즘을 설계하였다.

그리고, 기존의 다른 블록 암호 알고리즘과의 비교를 통해 NC 알고리즘의 안전성과 효율성 측면을 평가하였다. 특히, 키 생성 알고리즘에 의한 서브키 생성 속도가 2.42 μsec 정도로 매우 빠른 편이여서 MAC이나 해쉬 함수의 블록 구축시에 적용하면 상당한 효율성이 보장될 것이다.

제안한 NC 암호 알고리즘은 민간분야의 전자상거래 상에서의 안전성과 신뢰성을 보장하기 위한 목적으로 개발했으며, 개인 및 기업의 컴퓨터 내 중요 정보보호나 전자우편 시스템에서의 메시지 암호화, 그리고 인터넷을 이용한 전자상거래, 특히 전자화폐나 전자지불 시스템에 적합하게 사용될 수 있다.

또한, 금융망에서의 정보보호나 유료 수업 내용의 보호가 필요한 가상교육 시스템, 위성방송사업과 연계되어 유료 방송 내용에 관한 암호화를 필요로 하는 한정 수신 시스템(CAS: Conditional Access System) 등에 유용하게 사용될 수 있을 것이라 예측된다.

향후, NC 알고리즘의 보안 레벨을 좀 더 향상시키기 위한 방법의 하나로 암호화에 영향을 미치는 서브키를 견고하게 생성하기 위해 키 생성 알고리즘을 좀 더 보완하고, 동일한 구조로 설계된 다른 형태의 S-Box를 적용시켜 봄으로서 그것을 분석하고, 내부 함수 F 내부에 G 함수나 h 함수 같은 새로운 함수를 하나 더 추가하여 계산 복잡도를 증가시킴으로서 더욱 더 견고한 암호 알고리즘이 구축되리라고 생각된다.

참고 문헌

[1] A. Froomkin, "The Essential Role of Trusted Third Parties in Electronic Commerce", Ver. 1.02 Oct. 1996.

[2] N. Heintze, D. Tygar, "Model Checking Electronic Commerce Protocols", ARPA contract F33615-93-1-1330, Nov. 1995.

[3] R. Kailar, "Accountability in Electronic Commerce Protocols", IEEE Transaction on Software Engineering, Vol. 22, No. 5, May. 1996.

[4] O. Toole, "The Internet Billing Server Transaction Protocol Alternatives", Carnegie Mellon University Information Networking Institute, Apr. 1994.

[5] H. Feistel, "Cryptography and Computer Privacy", Scientific American, V. 228, N. 5, May 1973.

[6] H. Sun, "Computer and Network Security", Lecture by Rivest of Massachusetts Institute of Technology, http://theory.lcs.mit.edu/~rosario/6.915/lecture6.

[7] J. Daemen, L. Knudsen, V. Rijmen, "The Block Cipher Square", Fast Software Encryption, 4th International Workshop Proceedings, Springer-Verlag, 1997.

[8] M. Matsui, "New Block Encryption Algorithm MISTY", Proceedings of the 4th International Workshop of Fast Software Encryption, 1997.

[9] C. Adams, "Constructing Symmetric Ciphers Using the CAST Design Procedure", Designs, Codes and Cryptography, Vol. 12, No. 3, 1997.

[10] W. Madryga, "A High Performance Encryption Algorithm", Computer Security: A Global Challenge, Elsevier Science Publishers, 1984.

[11] K. Nyberg, "Linear Approximation of Block Ciphers", Advanced in Cryptology EUROCRYPT '94, LNCS 950, 1995.

[12] B. Preneel, "Analysis and Design Cryptographic Hash Functions", Ph.D., dissertation, Katholieke Universiteit Leuven, Jan. 1993.

[13] NIST, "Announcing Development of a Federal Information Standard for Advanced Encryption Standard", Federal Register, Vol. 62, No.1, Jan. 1997.

[14] B. Schneier, D. Whiting, "Fast Software Encryption: Designing Encryption Algorithms for Optimal Speed on the Intel Pentium Processor", Fast Software Encryption, 4th International Workshop Proceedings, Springer-Verlag, 1997.

[15] J. Daeman, "Cipher and Hash Function Design", Ph.D. thesis, Katholieke Universiteit Leuven, Mar. 1995.

저자 소개

서장원 (e-mail : jwsuh@duck.snut.ac.kr)
1992년 : 서울산업대학교 전산과(공학사)
1996년 : 숭실대학교 정보과학대학원 전산공학과(공학석사)
1998년 : 숭실대학교 대학원 컴퓨터학과 박사과정 수료
관심분야 : 암호 알고리즘, 전자지불 시스템, 네트워크 보안

전문석(e-mail : mjun@computing.soongsil.ac.kr)
1980년 : 숭실대학교 전산과(공학사)
1986년 : Univ. of Maryland 전산과(공학석사)
1988년 : Univ. of Maryland 전산과(공학박사)
1989년 : Morgan State Univ. 전산수학과 조교수
1991년 : New Maxico State Univ. 부설 Physical Science Lab. 책임연구원
1991년~현재 : 숭실대학교 컴퓨터학부 교수
관심분야 : 암호학, 병렬 알고리즘, 병렬 컴퓨터구조, 네트워크 이론