의료정보 보안을 위한 블록 암호 알고리즘의 설계

정 혜 명† · 전 문 석††

요 약

이 논문에서 제안한 MIT(Medical Information Transmission) 암호 알고리즘은 의료정보 전송을 위한 PACS의 표준 프로토콜인 DICOM표준을 위하여 설계되었다. 또한, 암호화에 민감한 영향을 미치는 요소 중에 하나인 서브키를 생성하기 위해 키 생성 알고리즘을 보다 복잡하게 설계함으로써 계산 복잡도의 증가와 확률 계산의 증가를 도모하였다.

Design of Block Cipher Algorithm for Medical Information Security

Hye-Myoung Choung† · Moon-Seog Jun††

ABSTRACT

One of these solutions is the implementation of strong cipher algorithm. MIT(Medical Information Transmission) cipher algorithm proposed in this paper is good for the security and is designed to accord with DICOM Standards that is PACS' Standards Protocol for medical information transmission. 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.

키워드 : 의료정보(Medical Information), 보안(Security), 알고리즘(Algorithm)

1. 서 론

인터넷 기술의 발전과 의무기록전산화시스템(EMR : Electronic Medical Record), 처방전달시스템(OCS : Order Communication System) 및 의료영상저장 및 전송시스템(PACS : Picture Archiving Communication System) 등의 확대 보급으로 인하여 의료정보들의 전송에 있어서도 편리한 시스템에 의존하려는 경향이 늘어나면서 이들 정보들의 보안문제가 크게 대두되고 있다. 이러한 보안상의 문제점을 해결하기 위한 대안으로 제시되고 있는 것 중의 한가지가 암호 알고리즘이다. 이에 필요한 핵심 요건 중에 하나가 안전성과 효율성 그리고 무엇보다도 의료정보의 특성을 고려해 볼 때 암·복호화 속도가 빠르고 기존의 의료정보 시스템들과 잘 호환될 수 있는 암호 알고리즘의 구축이라 할 수 있다.

이 논문에서 제안한 암호 알고리즘은 암호화 기법의 분류 중 메시지를 블록 단위로 분할하여 암·복호화하는 블록 암호 알고리즘이며, 내부처리의 단위는 PACS의 표준 프로토콜로 사용되고 있는 DICOM(Digital Image Communications in Medicine)표준과 같은 크기로 처리하도록 설계하였다. 이 MIT 암호 알고리즘은 기본적으로 비선형 변환과 선형 변환의 적절한 조합에 의해 설계되었으며, 알고리즘의 전체 구조는 데이터 블록의 좌·우 측면에 교대로 비선형 변환을 적용시키는 Feistel 구조를 적용하여 설계하였다. 이를 바탕으로 암호문의 생성 속도가 빠르게 진행되며, 외부 공격에도 대처할 수 있도록 설계하였다.

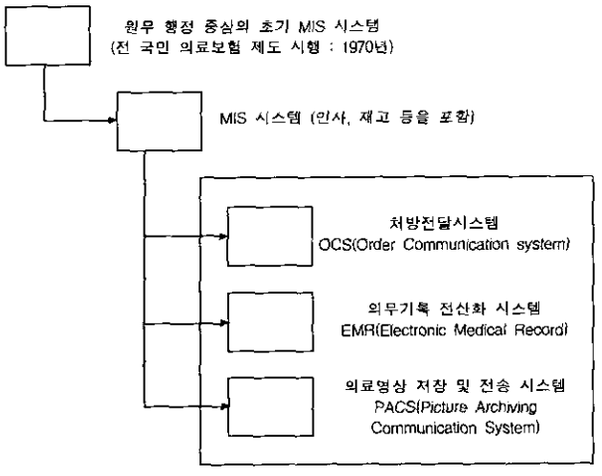
※ 이 논문은 2001학년도 김포대학의 연구비 지원에 의하여 연구되었음.
† 정 회 원 : 김포대학 컴퓨터계열 교수
†† 종신회원 : 숭실대학교 컴퓨터학부 교수
논문접수 : 2000년 12월 28일, 심사완료 : 2001년 5월 25일

2. 의료정보 및 관련 프로토콜과 블록 암호 알고리즘

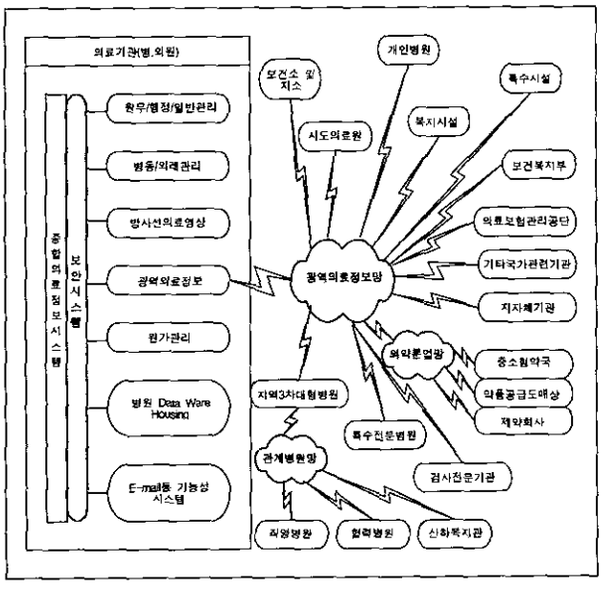
2.1 의료정보

의료정보라 함은 의사와 환자를 중심으로 또는 그와 연관되어 발생되는 모든 정보들을 말한다. 의료정보 전달 체제에서 데이터의 보안은 환자 개인의 프라이버시도 중요하지만 특별히 더 중요시하는 이유는 데이터의 완전성(Completeness) 정확성(Accuracy) 정밀성(Precision)이다. 최근 네트워크의 발달에 따른 개방형 통신망을 통하여 생명과 관계된 중요한 정보들이 손쉽게 상대방에게 전달된다. 즉 이러한 정보들을 다른 사람들이 손쉽게 정보에 접근 할 수 있다는 것을 의미한다. 따라서 생명과 관계된 데이터들에 관한 보안 문제는 더욱 더 중요한 문제가 아닐 수 없다. (그림 1)은 의료환경의 정보기술 도입에 관한 체계이고 (그림 2)는 의료정보 시스템 및 연관기관들과의 연계를 나타낸 그

팀이다[1].



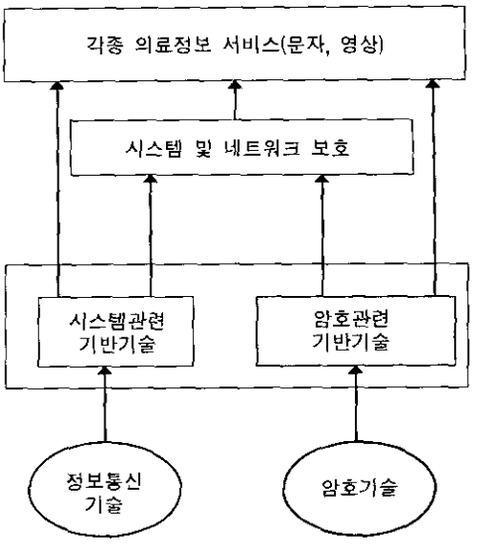
(그림 1) 의료환경의 정보기술 도입에 관한 체계



(그림 2) 의료정보 시스템 및 연관 기관과의 연계

2.2 의료정보 서비스와 정보보호

의료환경에서 볼 수 있는 컴퓨터 기반 환자기록에는 단순한 문자형의 정보에서부터 방사선 사진에 이르기까지 광범위하고 다양한 정보가 있다. 특히 이 분야에서는 진료 및 처방정보가 신속하고 정확하게 전달되어야 하는데 그러기 위해서는 통신매체도 중요하지만 그보다도 더 중요한 것은 정보의 안전성 즉 보안문제이다. 그러므로 이러한 보안 문제를 해결하기 위한 제반장치가 마련되지 않는다면 환자들의 중요한 정보들의 안전을 기대할 수 없게된다. 따라서 정보의 안전을 보장하기 위해서는 상대방을 확인함과 동시에 정보의 복제에 의한 정보의 부당한 누출이나 손상을 방지하는 대책이 필요하다[2]. (그림 3)은 의료정보 서비스와 정보보호 기술의 관계를 나타낸 것이다.



(그림 3) 의료정보 서비스와 정보보호

2.3 의료정보 관련 프로토콜

의료정보 관련 프로토콜로 PACS란 의료 영상 특히 방사선학적인 진단 영상들을 디지털 형태로 획득한 후, 고속의 통신망을 통하여 전송하고 과거의 X-ray 필름 보관 대신에 디지털 정보 형태로 의료 영상을 저장하며, 방사선과 의사들과 임상 의사들이 기존의 필름 뷰박스(Film Viewbox)대신에 영상조회 장치를 통하여 표시되는 영상을 이용하여 환자를 진료하는 포괄적인 디지털 영상 관리 및 전송 시스템을 말하며 PACS를 구현하기 위해서 영상 표시 및 처리, 정보통신 및 네트워킹, 데이터베이스, 정보관리, 사용자 인터페이스와 정보저장 관리 등의 기술들을 종합하여야 한다. 이러한 PACS시스템을 효과적으로 구축하기 위한 4대 원칙은 첫째, 임상적으로 수용 가능한 우수한 기술을 도입하고 유지하여야 하고 둘째, DICOM 표준 방식을 따라야 하며 셋째, 필름 없는 시스템을 경제적이고 효과적으로 만들고 유지하여야 하며 넷째, 병원정보 시스템의 기대에 부응하는 성능을 제공하여야 한다. 그래야만 PACS가 기존의 필름 중심의 업무보다 효과적으로 영상관리 업무를 향상시키며 환자진료의 질도 향상시킬 수 있다.

DICOM은 의료영상 장치들 사이에서 의료영상과 정보를 전송하는 업계표준 프로토콜로 미국 진단방사선과 협회와 의료장비업체간에 합의로 제안되었으며 유럽과 일본 그리고 우리나라 PACS학회와 의료장비업체들도 채택하고 있는 표준 프로토콜이다[3]. DICOM은 의료 영상을 교환하고 구성하는 방법과 그에 관련된 정보들을 기술한 자세한 명세(Specification)이다. DICOM은 산업표준 네트워크 연결을 사용하여 CT(Computed Tomography)와 MRI(Magnetic resonance imaging) 뿐만 아니라 핵의학, 초음파 등의 각종 디지털 영상 장비와 다른 정보 시스템간의 통신을 효과적으로 지원하며, 필름 프린터와 같은 영상 출력 장비와도 연결

하여 사용할 수 있도록 한다. 최근에는 대부분의 의료기 업체들이 DICOM표준을 수용하고 있는데 이는 의료 관련 기관들이 환자에 대한 서비스의 질을 향상시키고 의료 영상과 관련된 정보들을 다루는데 있어 통일성을 기하기 위함이다. DICOM프로토콜에서는 내부적으로 값 표현이 4바이트(32비트)씩으로 구성되어 있다[4].

Data Element Tag	Data Element Length	Data Element Value								
		First item			Second item			Third item		
(gggg, eeee) with VR of SQ	00000F0 0H	Item Tag (FFFF, E000)	Item Length (0000, 04F8H)	Item Value Data Set	Item Tag (FFFF, E000)	Item Length (0000, 04F8H)	Item Value Data Set	Item Tag (FFFF, E0000)	Item Length (0000, 04F8H)	Item Value Data Set
4 bytes	4 bytes	4 bytes	4 bytes	4 bytes	4 bytes	4 bytes	4 bytes	4 bytes	4 bytes	4 bytes

(그림 4) 3item을 갖는 DICOM 자료구조의 예

2.4 블록 암호 시스템

암호란 평문을 해독 불가능한 형태로 변형하거나 또는 암호화된 통신문을 해독이 가능한 형태로 변환하기 위한 원리, 수단, 방법 등을 다루는 기술이다. 암호화란 평문의 내용을 컴퓨터 상에서 XOR나 일련의 절차를 거치게 하여 변형시킨 암호문을 만드는 것을 말하고, 복호화란 이 암호문을 다시 평문으로 변형시키는 것을 말한다. 암·복호화를 시키는 일련의 과정을 암호 알고리즘이라 하고 이러한 암호 알고리즘은 사용되는 키의 특성에 따라 대칭 키 암호 알고리즘과 공개키 암호 알고리즘으로 나눈다. 암·복호화 키가 같은 암호 알고리즘은 메시지 처리 방식에 따라 블록 암호 알고리즘과 스트림 암호 알고리즘으로 나눌 수 있고 암·복호화 키가 서로 다른 공개키 암호 알고리즘은 키 공유 문제에 따라 공개키 분배 알고리즘과 공개키 관리 알고리즘으로 나눈다. 대칭 키 암호화 기법은 1972년부터 1974년 사이에 NIST(National Institute of Standard and Technology)가 암호 표준에 대한 문제를 처음 제기했고 그 결과가 세계적으로 가장 광범위하고 성공적으로 사용했던 DES 암호 알고리즘이었다. DES 암호화 기법은 1977년 IBM에서 개발하여 가장 광범위하게 사용되어온 미국 표준 암호 알고리즘으로, 메시지를 보내는 사람과 받는 사람이 동일한 키를 갖고 사용하는 개념으로써, 암호화되는 키가 작으므로 암호화가 빠른 장점이 있다. 현재 블록 암호 알고리즘 중에 가장 널리 알려져 있는 DES(Data Encryption Standard)는 키의 크기가 짧고 컴퓨터 속도의 개선과 암호해독 기술의 발전으로 오늘날 더 이상 DES를 안전하다고 생각하지 않게 되었다. 따라서 NIST에서는 DES를 공모했을 때와 같은 절차로 DES 알고리즘을 대체할 수 있는 새로운 알고리즘 AES(Advanced Encryption Standard)를 공모한 결과 15개국 21개회사에서 암호 알고리즘을 출품하여 그 중에서 1998년 15개가 1차 심사를 통과하였고 1999년 그 중 5개가 2차 심사를 통과하였으며 2000년 10월의 3차 심사에서 Rijndael 암호 알고리즘으로 결정되었으며 일정기간의 평가 기간을 거친 후 최종 결정될 예정이다.

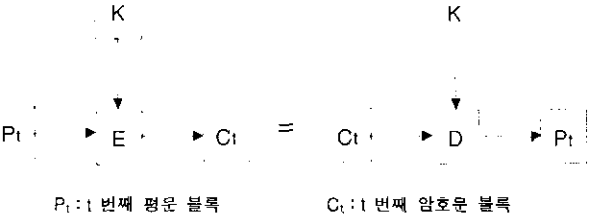
대부분 현대의 블록 암호 알고리즘은 "confusion과 diffusion의 반복에 의하여 강력한 암호 알고리즘을 설계할 수 있다"고 하는 Shanon의 이론을 기반으로 설계되었다[5, 8]. 즉, 혼돈(confusion)이론은 암호문 비트들의 통계적 분포가 평문 비트들의 통계적 분포에 어떻게 의존되는가를 판단하기 어렵게 만드는 것이고 확산(diffusion)이론은 평문의 각 비트들의 영향이 암호문 비트들에 어떻게 영향을 주는가를 판단하기 어렵게 만든다는 것이다. 따라서 이들 이론에 기초해서 만든 알고리즘들의 비도는 높아지는 장점이 있는 반면 단점으로는 블록 단위로 암호화가 이루어지므로 평문 비트들이 완전한 하나의 블록을 구성한 다음에 암호화의 과정이 이루어지므로 블록의 크기에 따라 지연 될 수 있으며 암호화 과정에서의 오류는 여러 변환에 영향을 미치므로 그 영향이 크다는 것이다. 블록 암호 시스템의 암호변환을 함수로 표현하면 다음과 같다.

$$C = f(P, K)$$

C는 암호문, P는 평문, K는 키, f는 합성 함수를 의미한다.

보통 n비트 블록 암호 알고리즘이란 고정된 n비트 평문이 동일한 크기를 갖는 n비트 암호문으로 바뀌는 것을 말하며 여기에서 말하는 n비트는 블록의 크기를 나타낸다. 이러한 변형 과정에서는 암호화 및 복호화 시 동일한 키가 사용된다[6]. 블록 암호 시스템은 크게 두 부분으로 나눌 수 있는데 한 부분은 암호 알고리즘부분 또 한 부분은 키 생성 부분으로 구성되며 실제로 암호 알고리즘은 공개되므로 비도는 키 생성부분에 의존해서 정해진다.

즉, 평문 블록의 크기를 n, 키 블록의 크기를 m이라 하면 2^n개의 가능한 평문 블록과 2^m개의 가능한 키 블록들이 존재한다. 함수 F_k를 키 K에 대한 암호화 함수라고 하면 이 함수는 치환이 되거나 크기가 n인 평문 블록과 암호문 블록 사이는 일대일 대응이 된다[7]. 대칭키 블록 암호의 모델을 보면 다음 (그림 5)와 같이 나타낼 수 있다.



(그림 5) 대칭키 블록 암호의 모델

(그림 5)에서 보면 암호문 블록 C_t와 평문 블록 P_t사이의 관계는 다음과 같이 나타낼 수 있다.

$$C_t = E(P_t, K) \qquad P_t = D(C_t, K)$$

블록의 크기가 짝수이고 다음과 같은 알고리즘을 가지는 암호를 h회전 Feistel 암호라고 한다.

- 블록의 크기 : $m = 2L$
- 평문블록 : $P = (x_0, x_1)$, 단 x_0와 x_1은 크기가 L인 블록
- 키 K가 부분 키 $k_1, k_2, \cdots k_n$으로 구성되고 각 부분 키 k_i에 대응하는 변환 $F_i : (Z_2)^L \rightarrow (Z_2)^L$이 정의되어 다음과 같은 암호화 과정을 거친다[6].

1 단계 : $P = (x_0, x_1) \rightarrow (x_1, x_2)$
2 단계 : $(x_1, x_2) \rightarrow (x_2, x_3)$
⋮ ⋮ ⋮
h 단계 : $(x_{h-1}, x_h) \rightarrow (x_h, x_{h+1}) = C$,
$x_{i+1} = x_{i-1} \oplus F_i(x_i)$

이때, 각각의 단계가 진행되는 과정을 회전이라고 한다. 이렇게 암호화된 암호문에 대한 복호화 과정은 암호문 $(x_h, x_{h+1}) = C$의 두 블록 x_h와 x_{h+1}의 순서를 바꾼 후 다음과 같이 h 단계를 통하여 얻은 두 블록 x_1과 x_0의 순서를 교환하면 평문 P를 다시 얻을 수 있다.

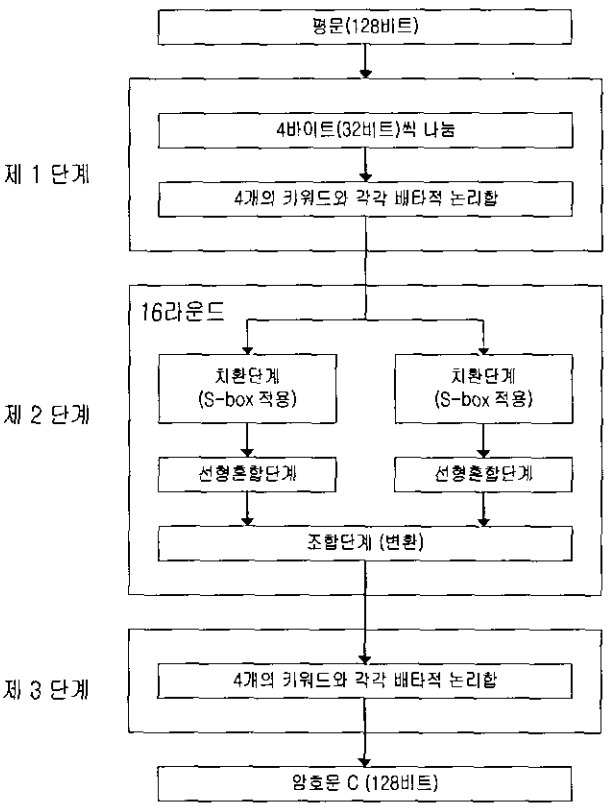
1 단계 : $P = (x_{h+1}, x_h) \rightarrow (x_h, x_{h-1})$
2 단계 : $(x_h, x_{h-1}) \rightarrow (x_{h-1}, x_{h-2})$
⋮ ⋮ ⋮
h 단계 : $(x_2, x_1) \rightarrow (x_1, x_0) = P$,
$x_{i-1} = x_{i+1} \oplus F_i(x_i)$

3. MIT 블록 암호 알고리즘

3.1 알고리즘 개요

이 논문에서 제안한 MIT 알고리즘은 대칭 키 암호 알고리즘에 바탕을 두고 고정된 크기의 입력 블록을 고정된 크기의 출력 블록으로 변형하는 즉 블록 단위로 메시지를 처리하는 블록 암호 알고리즘이다. 이 논문에서 제안한 MIT 알고리즘은 다음과 같이 설계되었다.

- 128비트 대칭형 블록 암호기이다.
- 128비트의 키크기를 갖는다.
- 알고리즘의 기본 구조는 Feistel구조를 이용한다.
- 라운드 수는 16라운드이다.
- 사용된 내부함수는 안전성이 입증된 치환테이블을 이용한다.
- 설계가 단순하고 분석과 구현이 쉽다.
- DICOM표준의 자료구조와 자료변환에 맞도록 내부처리의 단위를 4바이트(32비트)씩으로 하였다.



(그림 6) MIT 알고리즘의 구조

3.2 제 1단계

제 1단계 처리과정에서는 간단한 연산으로 평문을 4개의 32비트로 나누고 입력 단계에서 4개의 키워드와 배타적 논리합을 한다. 즉, 평문의 16바이트(128비트)는 처음에 4개의 4바이트(32비트)씩의 단어 p_0, p_1, p_2, p_3으로 나누어진다.

$$pi = \sum_{j=0}^{3} p(4i + j) \cdot 2^{8j} \qquad i = 0, 1, 2, 3$$

이들 단어들은 4개의 키워드와 배타적 논리합을 한다.

$$R_{0,i} = P_i \oplus K_i \qquad i = 0, 1, 2, 3$$

3.3 제 2단계

대부분의 블록 암호 알고리즘과 마찬가지로 제안하는 알고리즘도 Feistel 구조를 기반으로 설계하였고 Feistel 구조는 임의의 함수를 순열로 전환하는 일반적인 방법이며 보통 F 함수라고 부른다. 제 2단계에서는 S-boxes를 이용한 치환 단계, MDS 행렬을 이용한 선형혼합 단계, 변환을 이용한 조합 단계를 거친다. 이렇게 치환, 선형혼합, 조합의 세 단계를 한 라운드로 하여 16라운드를 거치게 된다. 두

워드씩이 F 함수의 입력으로 사용되고 나머지 두 워드는 앞의 라운드에서 나온 출력과 배타적 논리합을 취한 후 다음 라운드의 입력으로 사용된다. 따라서 다음과 같이 나타낼 수 있다.

$$(F_{r,0}, F_{r,1}) = F(R_{r,0}, R_{r,1}, r)$$
$$R_{r+1,0} = ROR(R_{r,2} \oplus F_{r,0}, 1)$$
$$R_{r+1,1} = ROL(R_{r,3}, 1) \oplus F_{r,1}$$
$$R_{r+1,2} = F_{r,0}$$
$$R_{r+1,3} = F_{r,1}$$

이때, r = 0, 1, …, 15 이고 ROR, ROL은 32비트의 입력변수를 회전시키는 함수이다. 제 2단계에서 4개의 S-box를 이용한 치환은 8비트의 입력을 취해서 8비트의 출력을 생성한다. 이때 사용한 S-box는 2개의 치환테이블을 순환적으로 적용하여 8*4씩 4개($S\text{-box}_0, S\text{-box}_1, S\text{-box}_2, S\text{-box}_3$)를 사용한다. 이 알고리즘에서 사용한 2개의 치환 테이블은 미분 공격을 피하고, 비 전사 처리단계 함수를 근간으로 하는 공격을 피하기 위해 만들어진 8*32의 크기를 갖고, 암·복호화 과정을 어렵게 하기 위해 키-의존 관계를 가지도록 설계되어 안전성이 입증된 치환테이블을 그대로 사용하였다[9, 10]. 키 종속 S-box를 갖는 암호기는 일반적으로 고정된 S-box보다는 안전하다고 믿을 수 있기 때문에 사용하였다. 즉, 미분 공격을 피하기 위하여 키 종속 S-Box를 사용하였다. 키에 종속되지 않은 S-box를 사용하면 키 설정 및 전체적으로 빠른 암호기를 구성할 수는 있지만 비도가 낮아진다.

S-box의 일반적인 설계원칙은 다음과 같다.

- 비선형이며 그것의 입력함수와 밀접하게 결합되지 않는다.
- 하나의 입력 비트는 결과적으로 적어도 두 개의 출력 비트를 교체한 S-box로 교체된다.
- 단일 비트가 일정하게 유지될 때 S-box출력에서 0과 1의 수 사이의 차이를 최소로 선택한다[10].

이러한 S-Box는 암호화에 민감한 영향을 미치므로 S-Box의 구성을 어떻게 하느냐에 따라 견고한 암호 알고리즘을 구축할 수 있다. 이 MIT 알고리즘에서는 Twofish의 S-box를 재사용 하였으므로 알려진 통계적 공격에 대하여 충분한 보호를 제공하고 있으며 유사한 미지의 공격에도 보호가 가능하며 비선형 대입 연산을 수행하는 비선형 함수로서, 입력 크기와 출력 크기를 변경시킬 수 있다. <표 1>, <표 2>는 MIT알고리즘에서 사용한 치환 테이블을 16 진수 표기법을 이용하여 개체들의 목록으로 표현한 것이다. 예를 들면 치환 테이블 1에서는 i = 01인 경우 즉 $S_1(01)$은 (10101001)을 16 진수 A9로 표현하고, 치환 테이블 2에서는 i = 01인 경우 즉 $S_2(01)$은 (01110101)을 16 진수 75로 표현한 것이다.

〈표 1〉 치환 테이블 1

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

MDS 행렬을 이용하는 선형혼합 단계에서는 앞의 치환의 결과, 즉 비 전사성을 가지는 S-box를 이용하여 8비트 입력을 취해서 8비트씩 출력된 4개의 결과는 4에 2^8승한 크기의 단일 벡터로 해석되며 고정계수를 갖는 4*4 MDS 행렬에 곱하여 진다. MDS 행렬은 a의 요소를 b의 요소로 선형사상 하면서 a+b 요소의 합성 벡터를 생성하면서 임의의 영이 아닌 값들에서 영이 아닌 요소의 최소의 수가 최소한 b+1을 갖는 성질이 있다. 즉 두 개의 구분되는 벡터 간의 거리(서로 다른 요소의 수)는 최장분리 가능 사상으로 생성되며 최소 b+1이다. 이 것은 두 개의 구분되는 벡터의 가장 작은 거리를 갖는 사상은 없다는 것이다. 최장 분리 가능 사상은 a*b 요소로 구성된 최장분리 가능 행렬로 표현된다. 제안한 알고리즘에서 사용한 MDS 행렬 또한 처리단계 함수 내에서 회전되자마자 바뀌는 바이트의 수를 보존하는 성질이 있고 고정 계수를 갖는다. 단지 세 개의 요소

01, EF, 5B를 갖는 이 MDS 행렬은 단일 바이트의 입력 편차에 의해서 출력 편차의 최소이진 *Hamming* 가중치를 최대로 할 수 있도록 선택된 것이다. 여기에서 MDS 행렬을 이용한 선형혼합 단계의 과정을 거치는 이유는 좋은 확산을 위해서이다. 이 선형 혼합 단계의 결과 벡터는 32비트 워드(8비트씩 4개)로 해석된다.

$$\begin{bmatrix} 01 & EF & 5B & 5B \\ 5B & EF & EF & 01 \\ EF & 5B & 01 & EF \\ EF & 01 & EF & 5B \end{bmatrix} \cdot \begin{bmatrix} y0 \\ y1 \\ y2 \\ y3 \end{bmatrix} = \begin{bmatrix} Z0 \\ Z1 \\ Z2 \\ Z3 \end{bmatrix}$$

↑

MDS 행렬

여기에서 y는 치환의 결과 값이고 Z는 MDS 행렬에 곱해진 결과이다. 현재 마이크로 프로세서 상에서 소프트웨어 구현을 위해서 MDS 행렬 곱셈은 보통 256개 32비트 워드로 각기 구성된 4개의 룩업 테이블로 구현되어진다. 따라서 처리 함수 단계 내에서 배타적 논리합 이후에 적용된 회전으로 인하여 회전이후 분산 성격을 보존하는 MDS 행렬의 선택이 요구된다. 암호화 및 복호화 양쪽 모

〈표 2〉 치환 테이블 2

i	$S_2(i)$	i	$S_2(i)$	i	$S_2(i)$	i	$S_2(i)$	i	$S_2(i)$	i	$S_2(i)$	i	$S_2(i)$	i	$S_2(i)$
00	75	01	F3	02	C6	03	F4	04	DB	05	7B	06	FB	07	C8
08	4A	09	D3	0A	E6	0B	6B	0C	45	0D	7D	0E	E8	0F	4B
10	D6	11	32	12	D8	13	FD	14	37	15	71	16	F1	17	E1
18	30	19	0F	1A	F8	1B	1B	1C	87	1D	FA	1E	06	1F	3F
20	5E	21	BA	22	AE	23	5B	24	8A	25	00	26	BC	27	9D
28	6D	29	C1	2A	B1	2B	0E	2C	80	2D	5D	2E	D2	2F	D5
30	A0	31	84	32	07	33	14	34	B5	35	90	36	2C	37	A3
38	B2	39	73	3A	4C	3B	54	3C	92	3D	74	3E	36	3F	51
40	38	41	B0	42	BD	43	5A	44	FC	45	60	46	62	47	96
48	6C	49	42	4A	F7	4B	10	4C	7C	4D	28	4E	27	4F	8C
50	13	51	95	52	9C	53	C7	54	24	55	46	56	3B	57	70
58	CA	59	E3	5A	85	5B	CB	5C	11	5D	D0	5E	93	5F	B8
60	A6	61	83	62	20	63	FF	64	9F	65	77	66	C3	67	CC
68	03	69	6F	6A	08	6B	BF	6C	40	6D	E7	6E	2B	6F	E2
70	79	71	0C	72	AA	73	82	74	41	75	3A	76	EA	77	B9
78	E4	79	9A	7A	A4	7B	97	7C	7E	7D	DA	7E	7A	7F	17
80	66	81	94	82	A1	83	1D	84	3D	85	F0	86	DE	87	B3
88	0B	89	72	8A	A7	8B	1C	8C	EF	8D	D1	8E	53	8F	3E
90	8F	91	33	92	26	93	5F	94	EC	95	76	96	2A	97	49
98	81	99	88	9A	EE	9B	21	9C	C4	9D	1A	9E	EB	9F	D9
A0	C5	A1	39	A2	99	A3	CD	A4	AD	A5	31	A6	8B	A7	01
A8	18	A9	23	AA	DD	AB	1F	AC	4E	AD	2D	AE	F9	AF	48
B0	4F	B1	F2	B2	65	B3	8E	B4	78	B5	5C	B6	58	B7	19
B8	8D	B9	E5	BA	98	BB	57	BC	67	BD	05	BE	05	BF	64
C0	AF	C1	63	C2	B6	C3	FE	C4	F5	C5	3C	C6	3C	C7	A5
C8	CE	C9	E9	CA	68	CB	44	CC	E0	CD	43	CE	43	CF	69
D0	29	D1	2E	D2	AC	D3	15	D4	59	D5	A8	D6	0A	D7	9E
D8	6E	D9	47	DA	DF	DB	34	DC	35	DD	6A	DE	CF	DF	DC
E0	22	E1	C9	E2	C0	E3	9B	E4	89	E5	D4	E6	ED	E7	AB
E8	12	E9	A2	EA	0D	EB	52	EC	BB	ED	02	EE	2F	EF	A9
F0	D7	F1	61	F2	1E	F3	B4	F4	50	F5	04	F6	F6	F7	C2
F8	16	F9	25	FA	86	FB	56	FC	55	FD	09	FE	BE	FF	91

두 배타적 논리합 이후에 한 비트의 오른쪽 회전이 일어난다. 이러한 회전 방향은 MDS 성격을 보존하도록 선택되어진 것이다.

조합 단계에서는 하나의 단순한 혼합동작으로 매우 빠르게 수행되어지며 주어진 두개의 입력 a와 b의 32비트 변환은 다음과 같이 선언한다.

$$a' = a + b \bmod 2^{32}$$
$$b' = a + 2b \bmod 2^{32}$$

이 단계는 서브 블록들과 키 사이의 혼돈을 위해서 수행되며 S-boxes를 이용한 치환 단계, MDS행렬을 이용한 선형혼합단계, 변환을 이용한 조합단계를 거친다.

3.4 제 3단계

제 2단계 이후에 마지막 처리 단계는 바꿈이 반대로 이루어지고 바꿈이 반대로 이루어진 다음 4개의 단어는 암호문을 생성하기 위하여 4개의 키 단어와 또 한번 배타적 논리합이 이루어지고, 128비트의 암호문이 생성된다.

$$C_i = R_{16,(i+2) \bmod 4} \oplus K_{i+4i} \quad i = 0, 1, 2, 3$$

암호문의 4개의 단어는 16바이트 C_0, C_1, …, C_{15}로 쓰인다.

$$C_i = \lfloor C_{\lfloor i/4 \rfloor} / 2^{8(i \bmod 4)} \rfloor \bmod 2^8 \quad i = 0, 1, \cdots, 15$$

3.5 키 생성

키 생성은 암호기가 사용하는 순환키로 전환되는 키 비트를 의미한다. 이 알고리즘에서는 많은 키 재료를 필요로 하며 복잡한 키 생성 단계를 거친다. 하지만 분석을 용이하게 하기 위해서 키 생성은 순환 함수를 이용한다. 서브키를 생성하는데 가장 중요한 설계목표는 안전성을 보장하는 것이다. 서브키는 암호 알고리즘에서 사용하는 순환키로 전환되는 키 비트를 의미한다. 키 생성과정 역시 128비트를 32비트씩 넷으로 나누고 이것을 또 한번 넷으로 나누어 8비트씩을 또 한번의 자리바꿈을 하여 서브키를 생성하도록 하였다. 키 K는 16바이트 즉 128비트로 구성되고 이것을 4바이트(32비트)씩 4개(K_0, K_1, K_2, K_3)의 워드로 나누어 처리한다. (그림 7)은 네 등분된 것 중의 하나인 32비트 K_0의 처리과정을 그림으로 나타낸 것이다.

첫 번째 단계에서는 32비트의 입력 단어를 8비트씩 4개로 M_0, M_1, M_2, M_3로 나누어 비 선형 함수 S-box로 치환하여 Y(Y_0, Y_1, Y_2, Y_3)를 생성한다.

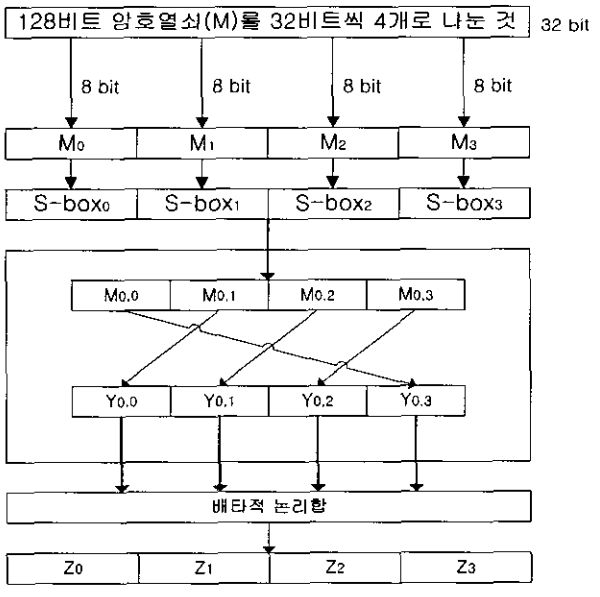
$$YS_0 = S_0(M_0),\ YS_1 = S_1(M_1),\ YS_2 = S_2(M_2),\ YS_3 = S_3(M_3)$$

두 번째 단계에서는 이들을 2비트씩 회전 이동시킨다. 여기에서 회전 이동을 시키는 이유는 이전 라운드의 키 값에 따라 몇 라운드 후에 키 값이 선형 변환 될 수 도 있으므

로 이를 방지하기 위함이다. 세 번째 단계에서는 이들을 XOR함으로써 $Z(Z_0, Z_1, Z_2, Z_3)$를 구한다.

$$Z_0 = (Y_0 \& M_{0,1}) \oplus (Y_1 \& M_{0,2}) \oplus (Y_2 \& M_{0,3}) \oplus (Y_3 \& M_{0,0})$$
$$Z_1 = (Y_0 \& M_{1,1}) \oplus (Y_1 \& M_{1,2}) \oplus (Y_2 \& M_{1,3}) \oplus (Y_3 \& M_{1,0})$$
$$Z_2 = (Y_0 \& M_{2,1}) \oplus (Y_1 \& M_{2,2}) \oplus (Y_2 \& M_{2,3}) \oplus (Y_3 \& M_{2,0})$$
$$Z_3 = (Y_0 \& M_{3,1}) \oplus (Y_1 \& M_{3,2}) \oplus (Y_2 \& M_{3,3}) \oplus (Y_3 \& M_{3,0})$$

따라서 $K_i(Z_{i,0}, Z_{i,1}, Z_{i,2}, Z_{i,3})$ i = 0, 1, 2, 3이다.



(그림 7) 서브키 생성 함수의 구조

이러한 구조는 기존의 Feistel 구조와 달리 이전 라운드의 출력 결과가 다음 라운드에 영향을 미치면서 그와 동시에 입력 블록과 배타적 논리합의 단계에서 변화를 일으켜 복잡도를 증가시키게 되므로 구조는 간단하고 알고리즘의 비도는 높아지게 되므로 좋은 구조라고 볼 수 있다.

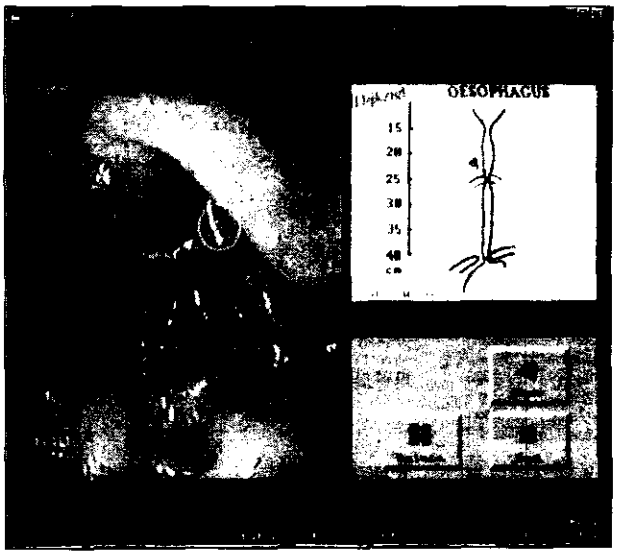
4. 구현 및 비교분석

4.1 구 현

MIT 알고리즘을 구현하기 위한 환경은 다음과 같다.

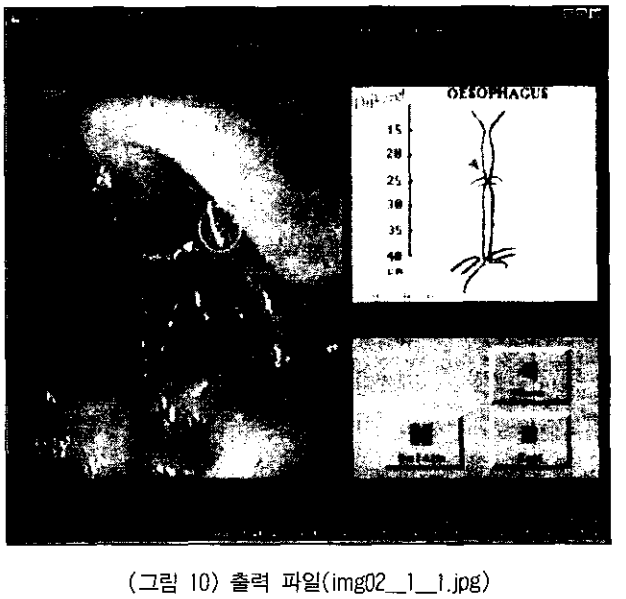
- 시스템 : Pentium Pro 200MHz
- 메모리 : 64MB
- OS : Windows 98
- 컴파일러 : MS Visual C/C++ 6.0

MIT 알고리즘에서 암호화 하고자하는 file을 받아들인다. 이 알고리즘에서의 암호화 과정은 평문과 암호문의 크기가 각각 128비트이고 암호키는 내부적으로 128비트를 취하여 키 생성 알고리즘에 의해 생성된다. 여기에서 입력으로 사용된 file은 img02_1.jpg 파일이다. 중간 파일은 암호화된 파일이고 복호화된 경과 파일은 img02_1_1.jpg 파일이다.



(그림 8) 입력 파일(img02_1.jpg)

(그림 9) 암호문 파일(img02_1.jpg파일의 암호문)



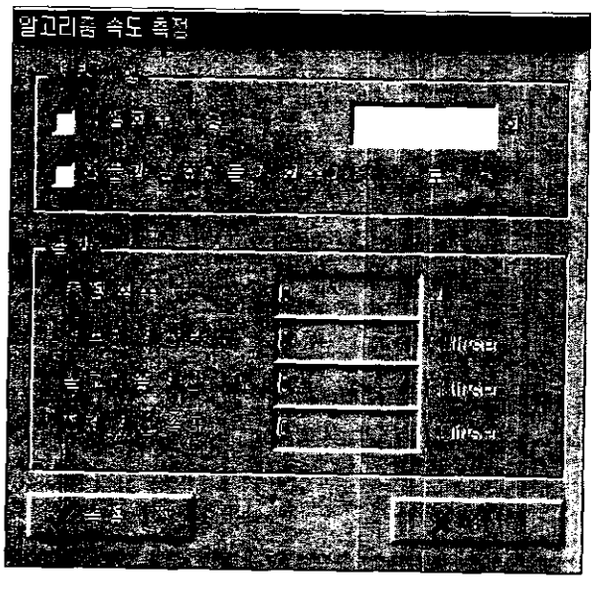
(그림 10) 출력 파일(img02_1_1.jpg)

4.2 비교분석

이 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 구조하에서 클럭(clock)에 비례하는 속도를 구하기가 용이하기 때문에 구조가 서로 다른 알고리즘을 평가하는 경우에는 매우 적합하게 사용되는 측정 방법이다[12]. 실험환경은 다음과 같다.

- 시스템 : Pentium II
- OS : Windows 98
- 컴파일러 : MS Visual C/C++ 6.0
- CPU : 200MHz
- 메모리 : 64MB



- 지정회수 사용 : 실험할 회수를 지정
- 평문과 암호키를 각 회수마다 난수로 입력 : 평문과 키 값을 난수와 고정 값으로 구분하여 선택
- 측정회수 : 실제 실험에 측정된 회수를 출력
- 서브키 계산 속도 : 키 생성 알고리즘에 의한 서브키 생성 속도를 출력
- 알고리즘 계산 속도 : 서브키를 제외한 알고리즘 자체의 속도를 출력
- 전체계산 속도 : 전체 암·복호화 수행속도 출력

그리고, 실험 범위는 다음과 같이 설정하였다.

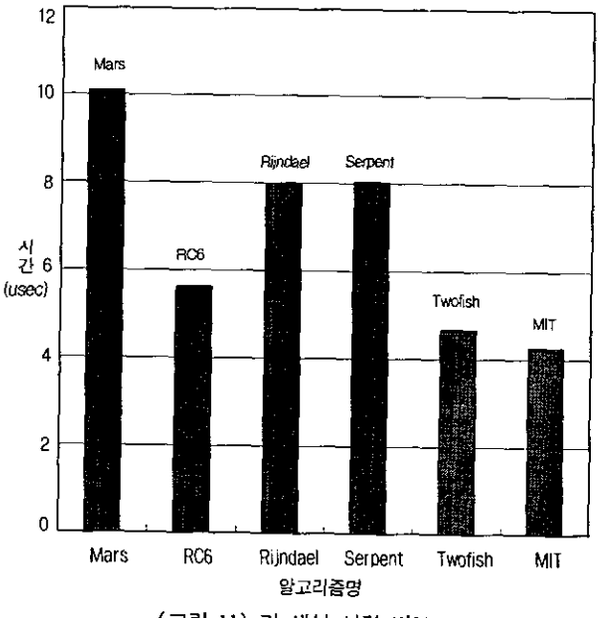
- 서브키 생성 시간은 임의의 키와 고정키를 사용하는 암호키에 대해 서브키 생성 알고리즘을 반복적으로 수행하여 이를 평균으로 산출한다.
- 암/복호화 처리 시간은 임의의 단위 블록을 반복적으로 암호화하는 과정과 이것을 반복적으로 복호화하는 과정을 수행하여 이를 평균으로 산출한다.
- 각 알고리즘에 대해 서브키 생성 시간과 암/복호화 처리 시간을 측정하기 위해 동일하게 10초간 수행하여 그 속도를 측정한다.

<표 3>은 위의 실험 환경과 실험 범위를 바탕으로 MIT 알고리즘과 기존의 암호 알고리즘들의 성능을 비교한 것이다. 위의 환경을 바탕으로 비교실험은 다음과 같이 기 개발된 알고리즘 측정 속도 툴을 이용하였다.

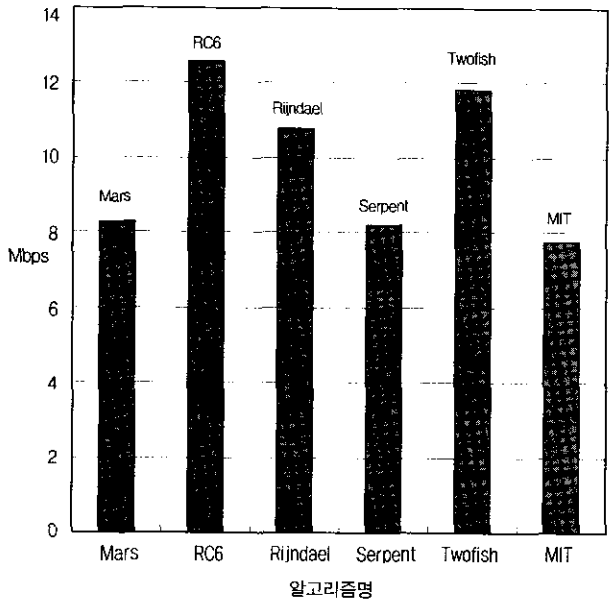
<표 3>은 알고리즘 속도 측정 툴에 의해 산출된 각각의 암호 알고리즘의 출력 중에서 가장 의미가 있다고 보는 서브키 계산 속도를 비교하여 정리한 것이다. 서브키 생성 속도는 각각의 암호 알고리즘의 키 생성 스케쥴링에 따른 생성속도에 초점을 맞추어 usec 단위로 환산한 것이다. 여기에서 암·복호화 처리부분은 각 암호 알고리즘의 암호문 산출 결과에 따른 실제 자료 처리량에 초점을 맞추어 Mbps단위로 환산한 결과를 나타낸 것이다. 여기에서 중요한 것은 암호화의 비도 및 안전성에 중요한 영향을 미치는 키 생성에 관한 부분을 중심으로 볼 때 제안한 MIT 알고리즘이 타 알고리즘에 비해 우세한 것으로 나타났다.

〈표 3〉 MIT알고리즘과 기존 암호 알고리즘들과의 성능비교

알고리즘명	키 생성 시간	암/복호화 처리
Mars	10.11 usec	8.27 Mbps
RC6	5.62 usec	12.56 Mbps
Rijndael	7.9 usec	10.78 Mbps
Serpent	8.03 usec	8.21 Mbps
Twofish	4.66 usec	11.82 Mbps
MIT	4.25 usec	7.75 Mbps



(그림 11) 키 생성 시간 비교



(그림 12) 암·복호화 처리 비교

(그림 11)은 각 알고리즘들의 키 생성 시간을 그래프로 비교해놓은 것이고, (그림 12)은 암/복호화 처리를 그래프로 비교해 놓은 것이다.

5. 결 론

이 논문에서는 의료정보 시스템에서 송·수신자간에 정보전송이 안전하게 이루어지게 하기 위해서는 이를 뒷받침해 줄 수 있는 장치가 필요한데, 그런 장치들 중에 하나가 바로 안전하고 효율적인 암호 알고리즘을 구축하는 것이라 할 수 있겠다.

이 논문에서 제안한 MIT 암호 알고리즘은 이러한 암호체계에 적합하도록 안전성과 효율성을 고려하였고 특히 기존의 의료정보 시스템에 잘 적용될 수 있도록 DICOM의 자료구조와 자료변환에 맞도록 내부처리를 4바이트씩 처리하도록 설계하였으며, 처리되는 블록 및 키의 크기는 128비트, 라운드 수는 16라운드, 알고리즘의 기본구조는 Feistel 구조를 이용하였으며, 내부함수에서 사용한 치환 테이블은 안전성이 입증된 것을 사용하였다.

또한, 일반적으로 암호화에 있어서 영향을 미치는 부분이 바로 암호키의 연산에 따라 이에 대응되는 키 비트 값이 결정되는 S-Box와 서브키를 생성하기 위한 키 생성 부분인데, 이를 위해 검증된 S-Box를 사용하였다. 이 MIT 암호 알고리즘이 특별히 의료정보를 암·복호화하는데 적합하다고 보는 이유는 크게 두가지이다. 첫째 MIT 알고리즘에서 평문을 32비트씩 처리하는 이유는 자료구조 및 자료변환 시 이를 4바이트(32비트)씩 처리하는 DICOM 표준과 일치하기 때문이고, 둘째는 의료정보가 문자자료보다는 영상자료가 더 많고 사이즈도 더 크다는 특성이 있으므로 대 용량의 자료를 암·복호화 하는데 이 논문에서 제안한 MIT 암호 알고리즘이 타 알고리즘에 비해 키 생성 속도 면에서 우수하기 때문이다. DICOM은 의료영상 장치들 사이에서 의료 영상과 정보들을 전송 및 저장하는 업계 표준 프로토콜로 PACS의 표준 프로토콜로 사용되고있으며, 인터넷 환경에서의 DICOM표준을 지원하는 실시간 의료정보 전송 및 저장기술에 관한 연구가 계속 진행되고 있는 실정이다[11].

앞으로 MIT 알고리즘의 보안 기능을 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 적용시켜서 계산 복잡도는 증가시키고 전체처리 속도를 향상시켜서 더욱 더 견고한 암호 알고리즘을 구축할 예정이다.

참 고 문 헌

[1] 대한의료정보학회, "보건의료정보학", 현문사, pp.201-227, 1999.

[2] 한국보건정보교육학회, "보건정보학개론", 현문사, pp.211-238, 2000.

[3] 김종효, 한만청, "병원정보시스템과 PACS의 통합", 대한PACS학회지, 제1권, pp.17-28, 1995.

[4] 박희정, 김선일, "한국형 PACS제원 표준안 제정을 위한 방향제시", 대한PACS학회지 제4권 제2호, pp.91-99, 1998.

[5] C. E. Shannon. Communication Theory of Secrecy System. Bell System Technical Journal, Vol.28, pp.656-715, October 1949.

[6] 이민섭, "현대 암호학", 교우사, 1999.

[7] 전자통신연구원, "암호학의 기초", 경문사, 1999.

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

[9] R. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," Communications of the ACM, Vol.21, No.2, Feb 1978, pp.120-126.

[10] E. Biham, A. Shamir, "Differential Cryptanalysis of DES-like Cryptosystem," Journal of Cryptology, Vol.4, No.1, 1991.

[11] National Electrical Manufactures Association, "Digital Imaging and Communications in Medicine," PS 3.5-1999, 1999.

[12] 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.

정 혜 명

e-mail : myoung@kimpo.ac.kr

1985년 고려대학교 간호학과(간호학사)

1995년 숭실대학교 정보과학대학원 전산공학과(공학석사)

1999년 숭실대학교 대학원 전산학과 박사과정 수료

1998년~현재 김포대학 컴퓨터계열 조교수

관심분야 : 병렬처리 시스템, 암호학 알고리즘, 인터넷 보안

전 문 석

e-mail : mjun@computing.soongsil.ac.kr

1981년 숭실대학교 전자계산학과(학사)

1986년 University of Maryland Computer Science(석사)

1988년 University of Maryland Computer Science(박사)

1989년 Morgan State Univ. 부설 Physical Science Lab. 책임연구원

1991년~현재 숭실대학교 컴퓨터학부 부교수

관심분야 : 병렬처리 시스템, 암호학 알고리즘, 인터넷 보안, 침입 차단 보안 시스템, 정보 이론, 전자상거래 보안