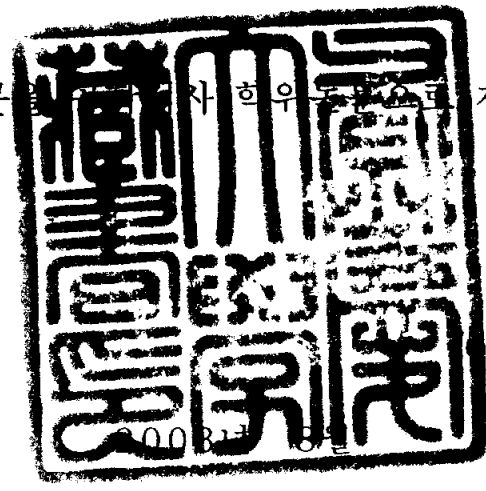
공학석사 학위논문

Substitution Permutation Networks에서 선형변환행렬의 MDS 코드 생성 판단 알고리즘

지도교수 조 경 연

이 논문을 공학석사 학위논문으로 제출함



부경대학교 산업대학원

컴 퓨 터 공 학 과

윤 성 훈

이 논문을 윤 성 훈의 공학석사 학위논문으로 인준함

2003년 6월 21일

주 심 공학박사 신 봉 기 

위 원 공학박사 권 오 흠 

위 원 공학박사 조 경 연 

차 례

Abstract iv

1. 서 론 1

2. 블록 암호알고리즘 3

2.1 AES(Advanced Encryption Standard) 5

2.1.1 암호화 과정 6

2.2 SEED 11

2.2.1 F 함수 11

2.2.2 G 함수 12

2.3 차분 공격과 선형 공격 13

2.3.1 차분 공격 13

2.3.2 선형 공격 14

2.4 치환계층과 교환계층의 특성 15

2.5 선형변환 행렬의 구성 방법 17

3. MDS 코드 생성확인 알고리즘 19

4. 구현 및 비교 검토 26

5. 결 론 30

참고문헌 31

그 림 차 례

그림 1 SP Network 3
그림 2 Feistal 구조 4
그림 3 AES의 암호화 과정 7
그림 4 AES의 아핀 변환 8
그림 5 SubBytes() 변환 8
그림 6 ShiftRows() 변환 9
그림 7 Mixcolumns() 변환 10
그림 8 AddRoundsKey() 변환 10
그림 9 F 함수 12
그림 10 G 함수 13
그림 11 연산 수 표시 그래프 28

표 차 례

표 1 키 길이와 블록 크기에 따른 라운드 수 6
표 2 기존 알고리즘의 연산 수 26
표 3 재귀호출 부분 27
표 4 변수를 소거하는 알고리즘의 연산 수 28

MDS Code Generation Confirmation Algorithms of Linear Transformation Matrix in Substitution Permutation Networks

Sung-Hun Yoon

Department of Computer Engineering

Graduate School of Industry

Pukyong National University

Abstract

The necessity of the information security has been increased on account of the development of the information communication service and the wide spread of the internet. Accordingly, a variety of cryptography algorithms is being developed and used in recent years. A high safety is kept using the skill of linear transformation in block cipher algorithm such as Substitution Permutation Network (SPN) among them. Linear and differential attacks show the strong characteristics against the attacks of block cipher algorithms; linear transformation matrix is characterized by the strength against these two attacks if it produces the Maximum Distance Separable(MDS) code.

In this paper, a new algorithm which can estimate that the linear transformation matrix produces the MDS code is proposed. First, input codes consist of over $GF(2^n)$ elements that interpret variable. Second, variable eliminate and estimate that generation MDS code of linear transformation matrix. The proposed algorithm reduces operation time greatly by decrease the number of multiplication and reciprocal operations compared with the conventional algorithm.

1. 서 론

정보통신의 발달과 인터넷의 확산으로 인하여 현대 산업사회 및 사회 일반에서 정보의 전달 수단에 중대한 변화가 일어났다. 이들 정보는 컴퓨터 시스템과 각종 통신 수단에 의하여 저장되고, 전달되게 되었으며, 이러한 과정에서 정보 보호의 필요성이 중요한 문제가 되었다. 그 결과 암호화 기술이 상당히 발전하였고 대중화되었다.

컴퓨터와 각종 전산망의 정보를 보호하기 위해서는 물리적으로 접근을 통제하는 것으로부터 비밀번호의 다단계 이용, 컴퓨터 운영 체계의 강화 등 많은 방법이 있으나, 정보의 직접적인 보호가 가장 기본적이고 안전한 수단이며 또한 최후의 방법이라 할 수 있다. 직접적인 정보보호 방법은 평이한 정보를 암호화된 정보로 만드는 것이다.

정보보호가 중요한 문제가 되자 미국 및 유럽을 비롯한 세계 각 국가들은 독자적으로 암호 알고리즘에 대한 연구를 하고 있으며, 암호기술 및 제품의 수출을 규제하고 있다. 이에 우리나라도 한국정보보호센터를 주축으로 연구하여 128비트 블록 암호 알고리즘인 SEED[1]를 개발하여 공개하였다.

암호화 기술이 발달함에 따라 암호문을 해독하기 위한 공격 기술도 같이 발달하게 되었으며, 자료의 암호화만으로는 더 이상 안전성을 보장할 수 없게 되었다. 이에 따라 메시지의 무결성, 사용자 인증, 전자서명과 같은 부가적인 기술이 필요하게 되었고, 공격에 대한 안전성을 고려한 여러 종류의 암호 알고리즘이 개발 활용되고 있다. 그 중에서 현재 많이 사용되고 있는 암호 알고리즘은 블록 암호 알고리즘이다.

블록 암호 알고리즘의 대표적인 공격 방법으로는 선형 공격(linear attack)[2]과 차분 공격(differential attack)[3]이 있다. Heys와 Tavares는 블록암호 알고리즘 가운데 하나인 치환-교환 네트워크(Substitution Permutation Networks : SPN)의

교환계층(permutation layer)으로 선형변환(linear transformation)을 사용하여, 암호문의 확산 특성이 개선되고, 선형과 차분 공격에 대한 저항성이 증가된다는 연구 결과를 발표하였다[4][5]. 선형변환은 수식적으로는 행렬로 표현할 수 있는데, 선형변환행렬이 Maximum Distance Separable(MDS) 코드[6]를 생성하면 암호 공격에 더욱 강하다는 것을 Vaudenay[7]가 제안하였고, SHARK[8] 및 SQUARE[9] 등의 블록 암호 알고리즘에 적용되었다.

본 논문에서는 SPN 구조를 가지는 블록 암호 알고리즘에서 교환계층의 선형변환행렬이 MDS 코드를 생성하는지 판단하는 새로운 알고리즘을 제안한다. 선형변환행렬의 입력코드는 $GF(2^n)$상의 원소들로 구성되며, 그 원소들을 변수로 해석하여, 변수를 소거시키면서 선형변환행렬이 MDS 코드를 생성하는지 여부를 판단한다. 기존 알고리즘인 선형변환행렬의 모든 정방부분행렬이 정칙인지를 확인하는 것과 비교하였을 때 계산 시간의 대부분을 차지하는 곱셈과 역수 연산 수가 크게 감소하였다.

본 논문에서 제안한 MDS 코드 생성 판단 알고리즘은 SPN 구조 및 SPN 구조로 근사할 수 있는 블록 암호 알고리즘의 치환계층 설계에 활용할 수 있을 것으로 기대된다.

본 논문의 구성은 2장에서는 SPN의 구조와 DES를 대체하여 새로운 표준 암호 알고리즘으로 채택된 AES[11], 한국정보보호센터에서 개발한 128 비트 블록 암호 알고리즘인 SEED, 블록 암호 알고리즘의 대표적 공격방법인 차분 공격과 선형 공격, SPN 구조를 가지는 블록 암호 알고리즘에서 치환계층과 교환계층의 특성 그리고 선형변환행렬의 구성 방법 등에 대해 알아보고, 3장에서 본 논문에서 제안한 알고리즘을 설명한 후, 4장에서 제안한 알고리즘의 구현 및 기존 알고리즘과의 성능을 비교 검토하고, 5장에서 결론을 맺는다.

2. 블록 암호 알고리즘

블록 암호는 고정된 크기의 입력 블록을 적당한 크기의 키를 이용하여 고정된 크기의 출력 블록으로 암호화하는 암호 시스템이다. 블록 암호에 있어서 출력 블록의 각 비트는 입력 블록과 키의 모든 비트에 의존하여 결정되도록 암호화하며, 입, 출력 블록의 크기는 구현상의 효율성과 보안상의 안전성에 크게 의존한다. 현재 쓰이는 대부분의 블록 암호들의 키 길이는 64, 80, 128, 256 비트이다.

대부분의 블록 암호는 라운드 함수를 반복적으로 적용하는 방법에 따라 크게 치환-교환 네트워크(Substitution Permutation Networks : SPN)와 피스탈 구조의 두 가지 형태로 구분한다. 그림 1과 그림 2에 각 구조의 형태를 나타내었다.

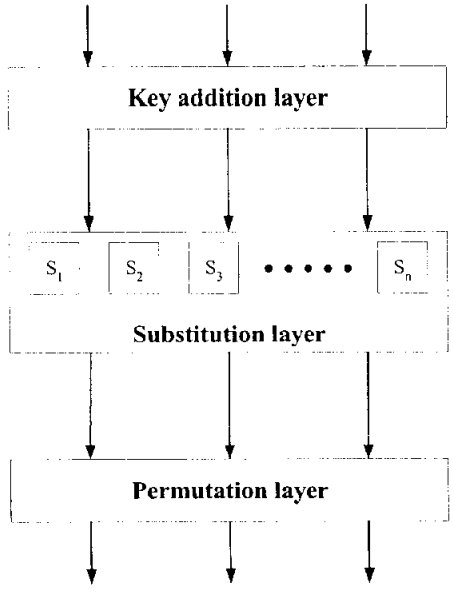


그림 1 SP Network

SPN은 여러 개의 함수를 중첩하면 개별 함수로 이루어진 암호보다 안전하

다는 이론에 근거하여 고전암호의 일종인 치환 암호와 교환 암호를 중첩하는 형태의 암호이다. 동작 방식은 입력을 여러 개의 소 블록으로 나누고 각 소 블록을 S 박스에 입력하여 치환시키고 S 박스의 출력을 P 박스로 교환하는 과정을 반복한다.

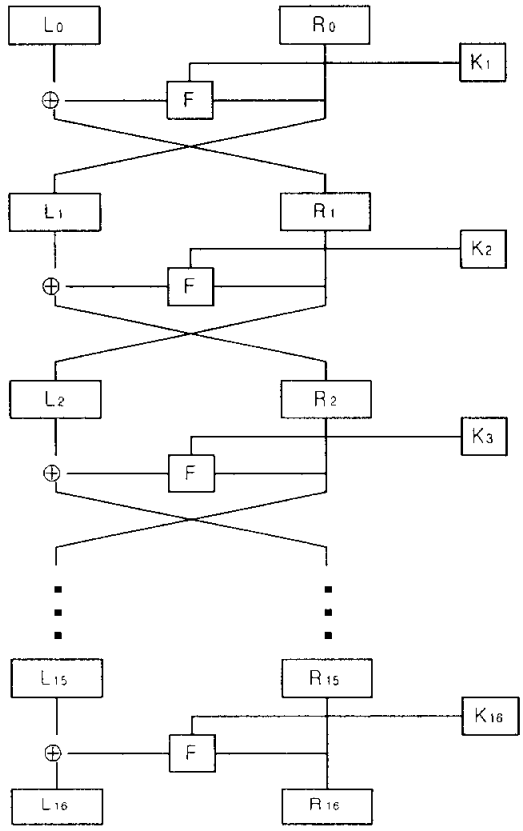


그림 2 Feistel 구조

피스탈 구조란 각각 t비트인 블록으로 이루어진 2t비트 평문 블록이 r라운드를 거쳐 암호문으로 변환되는 반복 구조를 말한다. 반복 구조란 평문 블록이 여러 라운드를 거쳐 암호화되는 과정을 말한다. 전체 알고리즘의 라운드 수는 요구되는 비도와 수행 효율성의 상호 절충적 관계에 의해 결정된다. 보통 피스

탈 구조는 3라운드 이상이며, 짝수 라운드로 구성된다. 이러한 피스탈 구조는 라운드 함수에 관계없이 암호화, 복호화 과정이 같은 역변환이 가능하고, 두 번의 수행으로 블록간의 완전한 확산이 이루어지며, 알고리즘의 수행속도가 빠르고, 하드웨어 및 소프트웨어로 구현이 용이한 장점을 지니고 있다.

2.1 AES(Advanced Encryption Standard)

AES는 미 상무부가 DES를 대체하기 위한 첨단암호표준(Advanced Encryption Standard)을 위한 알고리즘을 전 세계적으로 공모하여, 국립표준기술원(NIST)의 공개검사와 평가, 그리고 공개경쟁을 통하여 2000년 10월에 선정된 알고리즘이다. 개발자는 Joan Daemen과 Vincent Rijmen이고 보안, 성능, 효율성, 유연성이 최고로 결합된 암호표준이라는 점에서 선정됐다.

AES는 디자인 단계에서 다음의 세 가지 특징을 가졌다.

1. 모든 알려진 공격에 대한 저항력이 강하다.
2. 여러 플랫폼에서 속도가 빠르고 안정된 코드를 가진다.
3. 디자인이 단순하다.

AES의 한 라운드는 세 개의 계층으로 이루어진다.

1. Linear mixing layer : 여러 라운드에 걸친 높은 확산
2. Non-linear layer : 비선형 특성의 최적 조건을 가지는 S 박스의 병렬에 적용
3. Key addition layer : 라운드 키 연산

AES에서 블록은 128 비트로 고정이며 키는 128 비트, 192 비트 또는 256 비트를 가질 수 있다.

*Nb*를 평문 워드의 개수라 하고 *Nk*를 키워드의 개수라 하면 AES는 *Nb*와 *Nk*의 값에 따라 라운드 수가 변한다. 표 1에 변하는 라운드 수를 나타내었다.

표 1 키 길이와 블록 크기에 따른 라운드 수

	Key Length (*Nk words*)	Block Size (*Nb words*)	Number of Rounds (*Nr*)
AES-128	4	4	10
AES-196	6	4	12
AES-256	8	4	14

2.1.1 암호화 과정

AES의 전반적인 암호화 과정은 그림 3에 나타나 있다. 평문이 들어오면 먼저 AddRoundKey()가 동작되고, r－1라운드까지는 다음의 4개의 변환으로 구성되어 있다.

1. SubBytes() Transformation.
2. ShiftRows() Transformation.
3. MixColumns() Transformation.
4. AddroundKey() Transformation.

마지막 라운드는 MixColumns() 단계가 생략되어 SubBytes(), ShiftRows(), AddRoundKey()로 구성되어 있다.

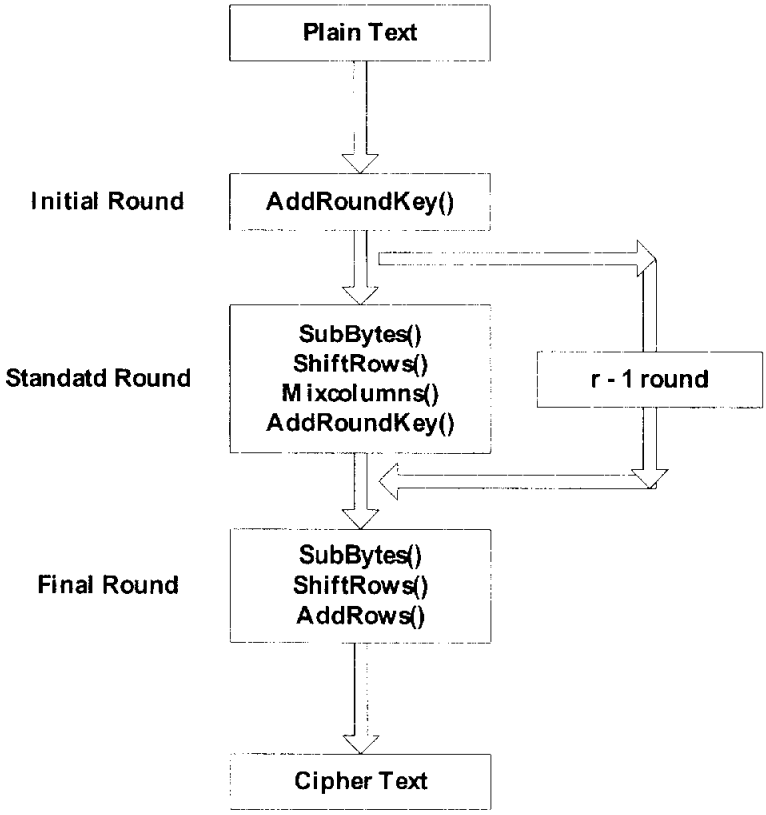


그림 3 AES의 암호화 과정

SubBytes() 변환은 S 박스에 해당하는 치환단계로, S 박스를 사용한 각각의 스테이트 바이트들이 독립적으로 비선형 바이트 치환을 한다. 입력을 각 바이트별로 원시 다항식에 의해 정의되는 유한체 $GF(2)$ 의 원소로 생각하여 역원에 대응시킨 후 그 값을 그림 4와 같은 아핀식에 의해 변환하여 얻는 단계이다.

$$\begin{bmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \\ b_4' \\ b_5' \\ b_6' \\ b_7' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

그림 4 AES의 아핀 변환

SubBytes() 변환을 그림으로 표현하면 그림 5와 같다.

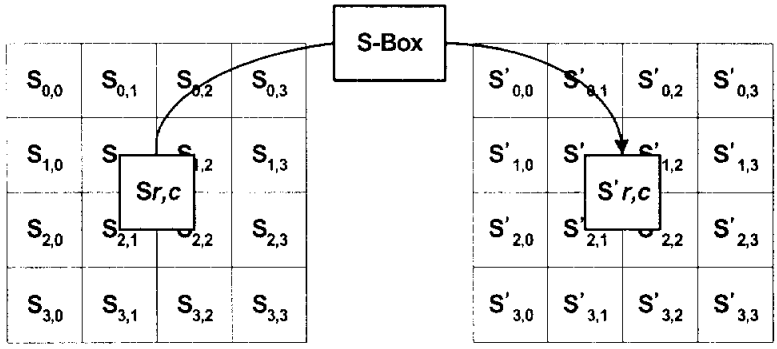


그림 5 SubBytes() 변환

ShiftRows() 변환은 $4\times Nb$ 행렬로 표현되는 입력을 각 행별로 순환 이동(rotation)시키는 단계이다. ShiftRows() 변환을 그림으로 표현하면 그림 6과 같다.

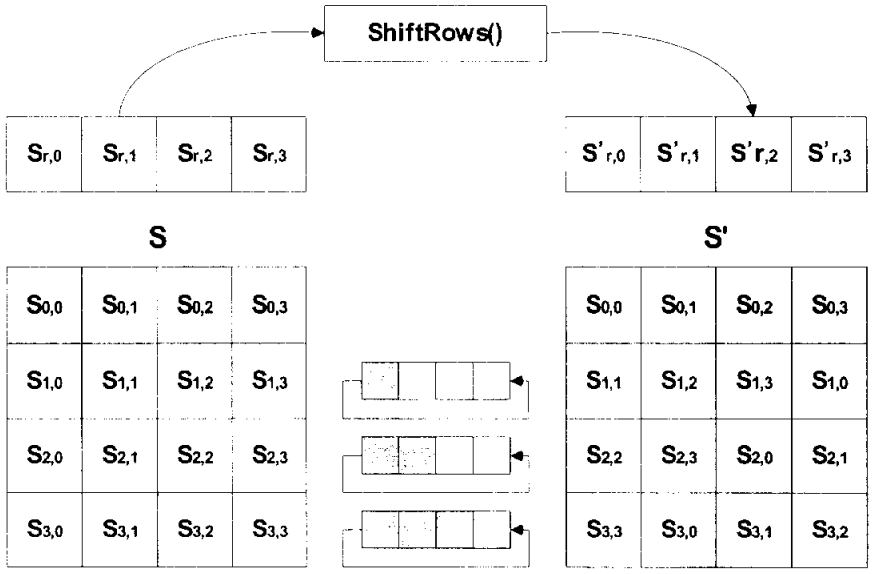


그림 6 ShiftRows() 변환

MixColumns() 변환은 입력 블록의 각 열에 해당하는 바이트 블록들이 서로 영향을 받도록 변환시켜 주는 단계로서 입력 열 블록과 출력 열 블록에 대하여 다음의 행렬식으로 표현되는 유한체 연산에 의해 변환된다.

$$\begin{bmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{2,c} \\ S'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \times \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix} \qquad \text{for } 0 \le c \le Nb$$

여기서 각 문자는 유한체 $GF(2)$의 원소를 나타내며 01, 02, 03은 각각 1, x, x+1을 나타낸다. MixColumns() 변환을 그림으로 표현하면 그림 7과 같다.

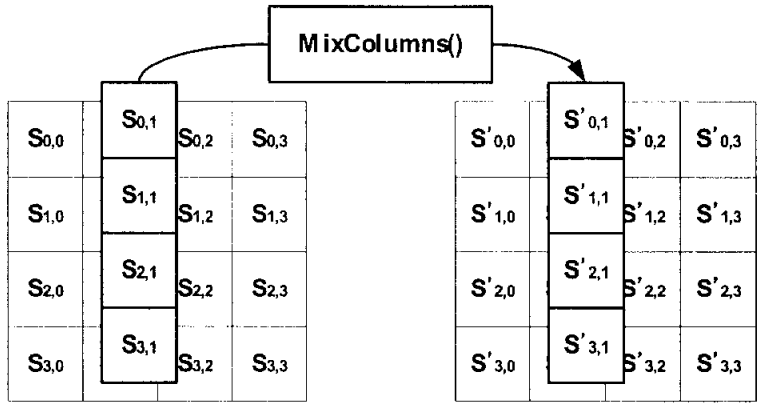


그림 7 MixColumns() 변환

AddroundKey() 변환은 State값 과 라운드 키 값을 XOR 연산으로 처리한다. AddroundKey() 변환을 그림 8에 나타내었다.

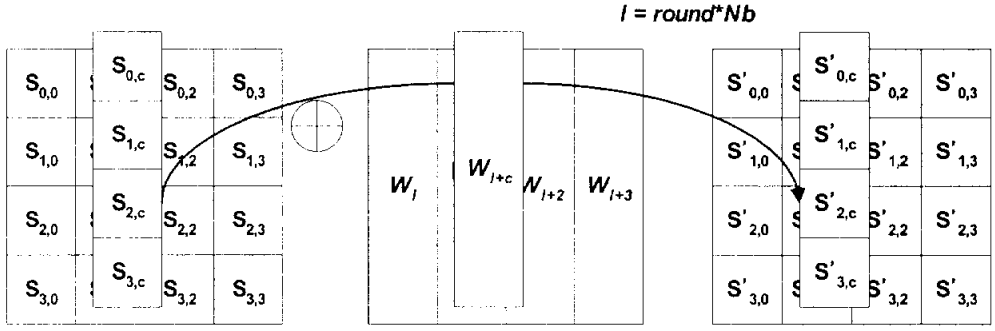


그림 8 AddRoundKey() 변환

2.2 SEED

SEED는 한국정보보호센터가 주축이 되어 개발한 대칭키 방식의 128비트 블록 암호 알고리즘이다.

SEED의 전체 구조는 피스탈 구조로 이루어져 있으며, 128비트의 평문 블록단위당 128비트 키로부터 생성된 16개의 64비트 라운드 키를 입력으로 사용하여 총 16라운드를 거쳐 128비트 암호문 블록을 출력한다.

SEED의 라운드 함수는 64비트를 입력받아 64비트 라운드 키로 변환하여 64비트를 출력하는데 내부의 G 함수를 3번 사용한다. G 함수는 차분 공격과 선형 공격에 대하여 안전성이 우수한 8비트 출력의 S 박스를 2개 사용하고, S 박스의 출력을 효과적으로 섞어 주는 선형 변환이 사용된다.

2.2.1 F 함수

SEED의 F 함수는 수정된 64 비트 피스탈 형태로 구성된다. F 함수는 각 32비트 블록 2개(C, D)를 입력으로 받아, 32비트 블록 2개(C', D')를 출력한다.

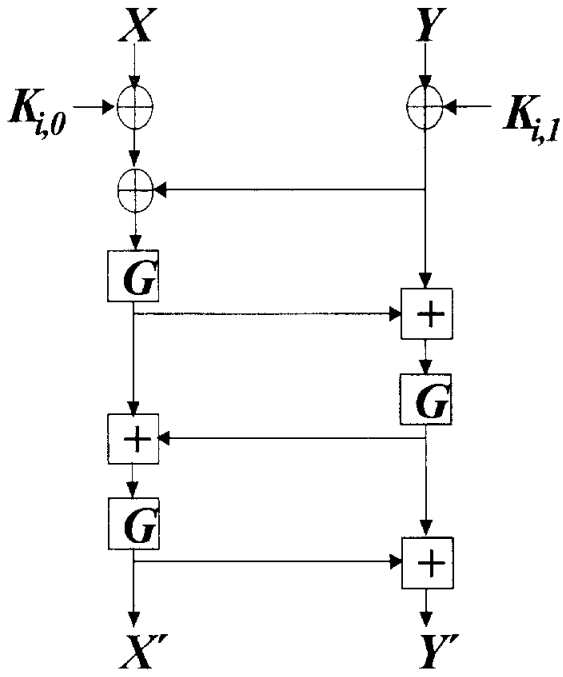


그림 9 F 함수

2.2.2 G 함수

전체 G 함수는 다음과 같다.

$$Y_3 = S_2(X_3),\quad Y_2 = S_1(X_2),\quad Y_1 = S_2(X_1),\quad Y_0 = S_1(X_0),$$
$$Z_3 = (Y_0 \,\&\, m_3) \oplus (Y_1 \,\&\, m_0) \oplus (Y_2 \,\&\, m_1) \oplus (Y_3 \,\&\, m_2)$$
$$Z_2 = (Y_0 \,\&\, m_2) \oplus (Y_1 \,\&\, m_3) \oplus (Y_2 \,\&\, m_0) \oplus (Y_3 \,\&\, m_1)$$
$$Z_1 = (Y_0 \,\&\, m_1) \oplus (Y_1 \,\&\, m_2) \oplus (Y_2 \,\&\, m_3) \oplus (Y_3 \,\&\, m_0)$$
$$Z_0 = (Y_0 \,\&\, m_0) \oplus (Y_1 \,\&\, m_1) \oplus (Y_2 \,\&\, m_2) \oplus (Y_3 \,\&\, m_3)$$

(단, m_0=0x*fc*, m_1=0x*f*3, m_2=0x*cf*, m_3=0x3*f* 이고, &는 bit-wise AND 연산을 의미한다)

G 함수의 그림을 그림 10에 나타내었다.

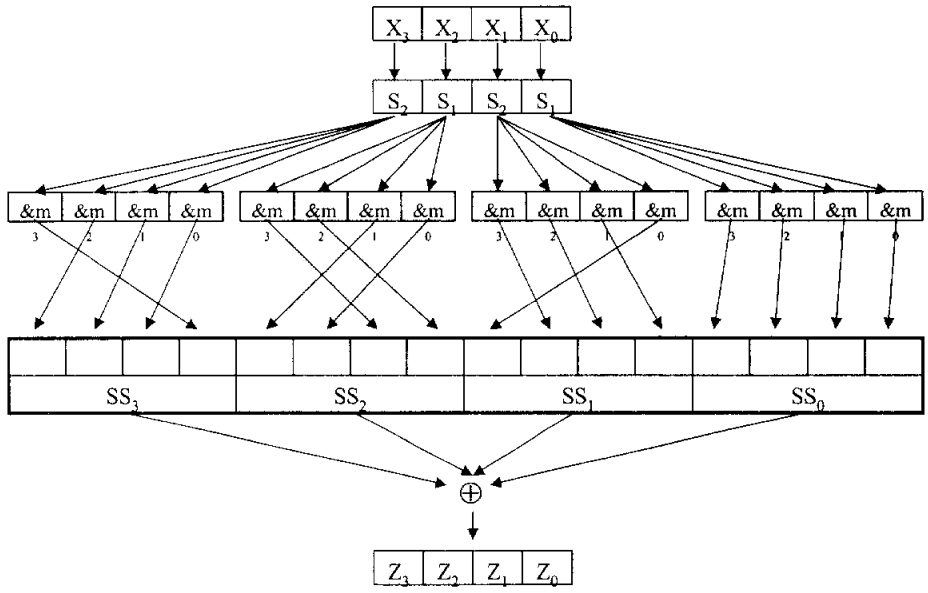


그림 10 G 함수

2.3 차분 공격과 선형 공격

2.3.1 차분 공격

차분 공격은 입력쌍의 차이와 해당 출력쌍의 차이 값들의 확률분포가 균일하지 않다는 사실을 이용하여 선택된 평문공격에 주로 사용되는 블록 암호 공격 방법이다. 차분 공격은 FEAL-4의 공격에서 Murphy에 의해 최초로 소개되었고[14,15], Biham과 Shamir 의해 완성되어, DES 공격에 성공하였다[2,16]. 차분 공격은 평문들이 같은 키를 가지고 암호화된 것처럼 연관된 두 개의 평문들 간의 차이점들을 전개한 것에 대한 분석을 신뢰한다. 유효한 데이터를 주의 깊게 분석한 것에 의하면 확률들은 각각의 가능한 키들로 할당될 수 있으며 결국 가

장 추정할 수 있는 키가 정정키로서 확인된다. 차분 공격은 성공의 단계들을 변화시킴으로서 대단히 많은 암호에 대하여 사용되어왔다. DES에 대한 공격에 있어 유효성은 1970년대 중반에 DES를 설계할 때 S박스들을 매우 조심스럽게 설계함으로서 제한되었다. 차분 공격에 대하여 암호를 보호하는 것에 대한 연구들은 Lai, Massey, Murphy 뿐 아니라 Nyberg와 Knudsen에 의해 진행되어 왔다[17,18]. 차분 공격은 또한 해쉬 함수들과 같이 다른 암호학적 알고리즘들의 공격에도 유용하게 사용되고 있다.

2.3.2 선형 공격

선형 공격은 확률적으로 근사시킨 비선형함수의 선형근사식을 이용하여 관련된 키비트를 추출하는 방법으로 알려진 평문공격에 이용된다. 선형 공격은 FEAL에 대한 공격으로 Matsui와 Yamagishi에 의해 처음으로 고안되었다[19,20]. DES를 공격하기 위해서 Matsui가 확장하였다[3]. 선형 공격은 알려진 평문 공격(known plaintext attack)이며 블록 암호의 특성을 설명하기 위해 선형 근사화를 이용한다. 평문과 그에 대응하는 암호문의 쌍들이 충분하게 주어졌을 때 키에 대한 정보의 비트들을 얻을 수 있으며 증가된 데이터 양들은 성공의 확률을 더 높여 준다. 기본적인 공격에 대해 다양하게 향상되어 왔다. Langford과 Hellman은 차분 선형 암호 공격이라고 불리는 공격을 연구하였다[21]. 이것은 차분 공격의 원소들과 선형 공격의 원소들을 결합한 것이다. 또한 Kaliski과 Robshaw는 다중 근사화들을 이용한 선형 공격은 요구되는 데이터 양을 감소시켰다[22].

2.4 치환계층과 교환계층의 특성

SPN 구조를 가지는 블록 암호 알고리즘의 한 라운드는 치환계층(Substitution layer), 교환계층(Permutation layer), 키 덧셈계층(Key addition layer)의 3계층으로 구성되어 있다. 선형 공격과 차분 공격에 강한 SPN 구조의 블록 암호 알고리즘의 치환 계층과 교환 계층에 대하여 강 주성 등[11]의 연구 결과가 있다.

치환계층은 비선형 특성을 가지는 복수개의 S 박스로 구성한다. 교환계층의 특성에 따라서 차분적 및 선형적으로 활동성을 가지는 S 박스의 수가 달라진다. 차분적으로 활동성을 가지는 S 박스는 입력 값이 '0'이 아닌 S 박스이며, 선형적으로 활동성을 가지는 S 박스는 입력과 출력 마스크 값이 '0'이 아닌 S 박스이다. 따라서 차분적 및 선형적으로 활동성을 가지는 S 박스의 최소수를 구하면

$$\beta_d(D) = \min_{\Delta x \neq 0} \{H_c(\Delta x) + H_c(\Delta y)\}$$

$$\beta_l(D) = \min_{b \neq 0} \{H_c(a) + H_c(b)\}$$

이고, 여기서 H는 해밍 가중치이다[11].

교환계층은 $GF(2^n)$상의 행렬 M으로 나타낼 수 있으며, 행렬의 출력과 입력 마스크 값 사이의 관계는 전치 행렬 M'로 나타낼 수 있다. 따라서 다음 식이 성립한다.

$$\Delta y = M\Delta x \ , \quad a = M^t b$$

이 행렬을 사용하면 차분적이고 선형적으로 활동하는 S 박스의 최소수를 각각 다음 식으로 계산할 수 있다.

$$\beta_d(D) = \min_{\Delta x \neq 0} \{H_c(\Delta x) + H_c(M\Delta x)\}$$

$$\beta_l(D) = \min_{b \neq 0} \{H_c(M^t b) + H_c(b)\}$$

위 식으로부터 행렬 M이 대칭행렬 혹은 직교행렬이면 차분적 및 선형적으로 활동성을 가지는 S 박스의 최소수는 동일하게 된다[11].

한편 교환 계층은 $GF(2^n)$상의 곱셈을 포함하고 있다. 행렬의 원소들은 $GF(2)$상의 행렬 곱셈으로 나타낼 수가 있으며, 이는 해밍 가중치로 표현할 수 있다. 해밍 가중치에 따라서 곱셈의 특성이 다르게 나타난다[13].

이들 연구로부터 차분 공격과 선형 공격에 강하기 위해서는 교환계층행렬이 대칭행렬 혹은 직교행렬이고, MDS 코드를 생성하고, 행렬의 각 원소들의 곱셈 특성은 [13]의 연구 결과를 충족시켜야 한다.

2.5 선형변환 행렬의 구성 방법

Youssef, Mister, Tavares은 두 가지 구성방법을 연구하였다[10].

***Lemma 2.1*[6]**

생성행렬 $G = [I|A]$를 가지는 어떤 (n, k, d)코드에서 A의 모든 정방부분행렬(임의의 i행과 i열로 구성되고, $i = 1, 2, \cdots, \min\{k, n-k\}$)이 정칙이면, 이 코드는 MDS이다. 여기서 A는 $k \times (n-k)$행렬이다.

이 방법은 선형변환행렬 A를 $GF(2^n)$상의 원소들로 임의로 구성한 후, ***Lemma 2.1***의 조건을 만족하는지 검사하는 방법으로 무작위 구성 방식이다. 이 방법으로 구성하면 행렬의 차수가 높아질수록 MDS 코드를 생성할 확률이 낮아진다.

***Lemma 2.2*[6]**

$x_0, \cdots, x_{n-1}$과 $y_0, \cdots, y_{n-1}$이 주어지고, 행렬 $A = [a_{ij}]$에서 $a_{ij} = 1/(x_i + y_j)$이고, 모든 i, j에 대해서 $x_i + y_j \neq 0$이면서, 다음 식을 만족하는 행렬을 Cauchy행렬이라고 부른다.

$$det(A) = \frac{\prod_{0 \le i < j \le n-2} (x_j - x_i)(y_j - x_i)}{\prod_{0 \le i,j \le n-1} (x_i + y_j)}$$

Cauchy행렬의 모든 정방부분행렬이 정칙이므로, MDS 코드를 생성한다. Cauchy 행렬은 행렬의 차수가 높을 때도 쉽게 구성할 수 있으나, 항상 특정한

원소들로만 구성되기 때문에 2.4절에서 설명한 치환 계층과 교환 계층의 특성을 만족하도록 선형변환행렬을 구성할 수가 없다. 따라서 행렬을 구성할 때는 임의 구성 방법으로 구성을 하여야한다. 임의 구성 방법으로 행렬을 구성한 경우에는 행렬의 차수가 높아질수록 MDS 코드를 생성할 확률이 낮아지게 된다. 그래서 MDS 코드 생성을 확인하는 알고리즘이 필요하다.

3. MDS 코드 생성확인 알고리즘

블록 암호 알고리즘에서 교환 계층은 $GF(2^n)$상의 M개의 n 비트 코드 $\boldsymbol{X}$를 M개의 n 비트 코드 $\boldsymbol{Y}$로 선형변환한다. 이 때 선형변환행렬을 $\boldsymbol{A}$라 하면 선형변환은 식(1)로 표현된다.

$$\begin{bmatrix} Y_0 \\ Y_1 \\ Y_2 \\ \cdots \\ Y_{M-1} \end{bmatrix} = \begin{bmatrix} A_{0,0} & A_{0,1} & A_{0,2} & \cdots & A_{0,M-1} \\ A_{1,0} & A_{1,1} & A_{1,2} & \cdots & A_{1,M-1} \\ A_{2,0} & A_{2,1} & A_{2,2} & \cdots & A_{2,M-1} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ A_{M-1,0} & A_{M-1,1} & A_{M-1,2} & \cdots & A_{M-1,M-1} \end{bmatrix} \times \begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ \cdots \\ X_{M-1} \end{bmatrix} \qquad (1)$$

식(1)에서 선형변환의 입력코드 $\boldsymbol{X}$의 해밍 가중치와 선형변환 후의 출력코드 $\boldsymbol{Y}$의 해밍 가중치를 각각 $W_h(\boldsymbol{X})$, $W_h(\boldsymbol{Y})$라 하면, 선형변환행렬 $\boldsymbol{A}$의 branch number 는 식(2)와 같이 주어진다.

$$\beta = \min(W_h(X) + W_h(Y)) \qquad while \quad X \neq 0 \qquad (2)$$

branch number $\beta = M + 1$이 되는 선형변환행렬 $\boldsymbol{A}$는 MDS 코드를 생성한다고 한다. 따라서 선형변환행렬 $\boldsymbol{A}$가 MDS 코드를 생성하는지 판단하기 위해서는 입력코드 $\boldsymbol{X}$의 모든 원소에 대하여 branch number $\beta = M + 1$을 만족하는지 확인해야 한다. 이를 위해서 본 논문에서는 입력코드 $\boldsymbol{X}$의 원소 $X_i(0 \leq i \leq M-1)$를 변수로 취급하여 이들을 소거하면서 연산하여, MDS 코드를 생성하는지 판단하는 알고리즘을 제안한다.

Lemma 3.1

M개의 원소를 가진 입력코드 $\boldsymbol{X}$의 해밍 가중치 '$W_h(\boldsymbol{X})$ = 1'이면, 선형변환 행렬 $\boldsymbol{A}$의 모든 원소가 '0'이 아니면 MDS 코드를 생성한다.

증명

입력코드 $\boldsymbol{X}$의 해밍 가중치 '$W_h(\boldsymbol{X})$ = 1'이므로 $\boldsymbol{X}$의 원소중 하나만이 '0'이 아니다. 그 원소를 $X_i(0 \leq i \leq M-1)$라 하면 식(1)은 식(3)과 같이 된다.

$$\begin{aligned} Y_0 &= A_{0,i} \times X_i \\ Y_1 &= A_{1,i} \times X_i \\ Y_2 &= A_{2,i} \times X_i \\ &\cdots\cdots \\ Y_{M-1} &= A_{M-1,i} \times X_i \end{aligned} \tag{3}$$

식(3)에서 $X_i \neq 0$이므로 선형변환행렬 $\boldsymbol{A}$의 모든 원소가 '0'이 아니면 출력코드 $\boldsymbol{Y}$의 모든 원소는 '0'이 아니고, 해밍 가중치 '$W_h(\boldsymbol{Y}) = M$'이 된다. 따라서 선형변환행렬 $\boldsymbol{A}$의 branch number $\beta = M + 1$이 되어 MDS 코드를 생성한다. □

Lemma 3.2

입력코드 $\boldsymbol{X}$의 해밍 가중치 '$W_h(\boldsymbol{X})$ = 2'이면, 출력코드 $\boldsymbol{Y}$의 해밍 가중치 최소값 '$W_h(\boldsymbol{Y}) = M-1$'인 경우에 MDS 코드를 생성한다. 입력코드의 원소 가운데 두 개의 원소만이 '0'이 아니고 그 원소를 X_i, X_j라 하면, 출력코드 원소 가운데 $Y_p(0 \leq p \leq M-2)$ 원소를 '0'으로 만드는 X_j가 반드시 존재한다. X_j를 X_i의 함수로 표현하고, 이것을 $Y_q(p < q \leq M-1)$에 대입하여 '$Y_q \neq 0$'가 되면 MDS 코드를 생성한다.

증명

입력코드 $\boldsymbol{X}$의 해밍 가중치 '$W_h(\boldsymbol{X}) = 2$'이므로 $\boldsymbol{X}$의 원소 가운데 두개만이 0이 아니다. 그 원소를 X_i, X_j라 두면 식(1)은 식(4)와 같이 표현된다.

$$\begin{aligned} Y_0 &= (A_{0,i} \times X_i) + (A_{0,j} \times X_j) \\ Y_1 &= (A_{1,i} \times X_i) + (A_{1,j} \times X_j) \\ Y_2 &= (A_{2,i} \times X_i) + (A_{2,j} \times X_j) \\ &\cdots\cdots\cdots \\ Y_{M-1} &= (A_{M-1,i} \times X_i) + (A_{M-1,j} \times X_j) \end{aligned} \tag{4}$$

식(4)에서 X_j를 소거시키기 위하여 '$Y_0 = 0$'으로 가정하고, X_j에 대해서 정리하면 식(5)가 된다.

$$X_j = -\frac{A_{0,i} \times X_i}{A_{0,j}} \tag{5}$$

식(5)를 식(4)에 대입하여 정리하면 식(6)과 같이 된다.

$$\begin{aligned} Y_1 &= \left(A_{1,i} - \frac{A_{1,j} \times A_{0,i}}{A_{0,j}}\right) \times X_i \\ Y_2 &= \left(A_{2,i} - \frac{A_{2,j} \times A_{0,i}}{A_{0,j}}\right) \times X_i \\ Y_3 &= \left(A_{3,i} - \frac{A_{3,j} \times A_{0,i}}{A_{0,j}}\right) \times X_i \\ &\cdots\cdots\cdots \\ Y_{M-1} &= \left(A_{M-1,i} - \frac{A_{M-1,j} \times A_{0,i}}{A_{0,j}}\right) \times X_i \end{aligned} \tag{6}$$

식(6)에서 출력코드 Y_1에서 Y_{M-1}까지의 모든 값이 '0'이 아니면 Y의 해밍 가중치의 최소값은 '$W_h(\boldsymbol{Y}) = M-1$'이다.

식(4)에서 $Y_i = 0$이라고 가정하면, 앞에서 $Y_0 = 0$이면서 $Y_1 = 0$인 경우가 없음을 확인하였으므로 $Y_0 \neq 0$이다. 한편 X_j는 식(7)이 된다.

$$X_j = -\frac{A_{1,i} \times X_i}{A_{1,j}} \tag{7}$$

Xj를 소거하기 위하여 식(7)을 식(4)에 대입하여 정리를 하면 식(8)과 같이 된다.

$$\begin{aligned}
Y_2 &= \left(A_{2,i} - \frac{A_{2,j} \times A_{1,i}}{A_{1,j}}\right) \times X_i \\
Y_3 &= \left(A_{3,i} - \frac{A_{3,j} \times A_{1,i}}{A_{1,j}}\right) \times X_i \\
Y_4 &= \left(A_{4,i} - \frac{A_{4,j} \times A_{1,i}}{A_{1,j}}\right) \times X_i \\
&\cdots\cdots\cdots \\
Y_{M-1} &= \left(A_{M-1,i} - \frac{A_{M-1,j} \times A_{1,i}}{A_{1,j}}\right) \times X_i
\end{aligned} \tag{8}$$

식(8)에서 출력코드 값인 Y_2에서 Y_{M-1}이 모두 '0'이 아니면, $Y_0 \neq 0$이므로 $\boldsymbol{Y}$의 해밍 가중치의 최소값은 '$W_h(Y) = M-1$'이 된다.

이와 같은 과정을 $Y_i(2 \leq i \leq M-2)$에 대하여 반복하여 모든 출력 코드 값 Y_{i+1}에서 Y_{M-1}이 모두 '0'이 아니면, Y의 해밍 가중치 최소값은 '$W_h(\boldsymbol{Y}) = M-1$'이 된다. □

Lemma 3.3

입력코드 $\boldsymbol{X}$의 해밍 가중치 '$W_h(\boldsymbol{X}) = 3$'이면, 출력코드 Y의 해밍 가중치 '$W_h(\boldsymbol{Y}) = M-2$'인 경우에 MDS 코드를 생성한다. 입력코드 $\boldsymbol{X}$의 원소 가운데 세 개만이 0이 아니고 그 원소를 X_i, X_j 그리고 X_k라 하면, 출력코드의 원소 $Y_p(0$

$\leq p \leq M-3$)를 '0'으로 가정하여 X_k를 X_i 와 X_j의 함수로 표현하고, 이것을 $Y_q(p < q \leq M-2)$에 대입하여 X_k를 소거한다. Y_q는 '$W_h(\boldsymbol{X}) = 2$'인 $M-q$ 차원의 행렬이 되며, Y_q가 ***Lemma*** 3.2를 만족하면 선형변환행렬은 MDS 코드를 생성한다.

증명

입력코드 $\boldsymbol{X}$의 해밍 가중치 '$W_h(\boldsymbol{X}) = 3$'이므로 $\boldsymbol{X}$의 원소 가운데 세 개만이 '0'이 아니다. 그 원소를 X_i, X_j 그리고 X_k 라 두면 식(1)은 식(9)와 같이 된다.

$$\begin{aligned} Y_0 &= (A_{0,i} \times X_i) + (A_{0,j} \times X_j) + (A_{0,k} \times X_k) \\ Y_1 &= (A_{1,i} \times X_i) + (A_{1,j} \times X_j) + (A_{1,k} \times X_k) \\ Y_2 &= (A_{2,i} \times X_i) + (A_{2,j} \times X_j) + (A_{2,k} \times X_k) \\ &\cdots\cdots\cdots\cdots \\ Y_{M-1} &= (A_{M-1,i} \times X_i) + (A_{M-1,j} \times X_j) + (A_{M-1,k} \times X_k) \end{aligned} \tag{9}$$

식(9)에서 X_k를 소거시키기 위하여 $Y_0 = 0$이라고 가정하고 X_k에 대해서 정리하면 식(10)이 된다.

$$\begin{aligned} Y_1 &= \left\{\left(A_{1,i} - \frac{A_{0,i} \times A_{1,k}}{A_{0,k}}\right) \times X_i\right\} + \left\{\left(A_{1,j} - \frac{A_{0,j} \times A_{1,k}}{A_{0,k}}\right) \times X_j\right\} \\ Y_2 &= \left\{\left(A_{2,k} - \frac{A_{0,i} \times A_{2,k}}{A_{0,k}}\right) \times X_i\right\} + \left\{\left(A_{2,j} - \frac{A_{0,j} \times A_{2,k}}{A_{0,k}}\right) \times X_j\right\} \\ Y_3 &= \left\{\left(A_{3,k} - \frac{A_{0,i} \times A_{3,k}}{A_{0,k}}\right) \times X_i\right\} + \left\{\left(A_{3,j} - \frac{A_{0,j} \times A_{3,k}}{A_{0,k}}\right) \times X_j\right\} \\ &\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ Y_{M-1} &= \left\{\left(A_{M-1,i} - \frac{A_{0,i} \times A_{M-1,k}}{A_{0,k}}\right) \times X_i\right\} + \left\{\left(A_{M-1,j} - \frac{A_{0,j} \times A_{M-1,k}}{A_{0,k}}\right) \times X_j\right\} \end{aligned} \tag{10}$$

여기서 식을 간소화하기 위하여

$$\left(A_{1,i}-\frac{A_{0,i}\times A_{1,k}}{A_{0,k}}\right)=A'_{1,i},\quad \left(A_{1,j}-\frac{A_{0,j}\times A_{1,k}}{A_{0,k}}\right)=A'_{1,j},\quad \cdots,$$

$$\left(A_{M-1,i}-\frac{A_{0,i}\times A_{M-1,k}}{A_{0,k}}\right)=A'_{M-1,i},\quad \left(A_{M-1,j}-\frac{A_{0,i}\times A_{M-1,k}}{A_{0,k}}\right)=A'_{M-1,j}$$

로 치환하면 식(10)은 식(11)로 나타낼 수 있다.

$$\begin{aligned} Y_1&=(A'_{1,i}\times X_i)+(A'_{1,j}\times X_j)\\ Y_2&=(A'_{2,i}\times X_i)+(A'_{2,j}\times X_j)\\ Y_3&=(A'_{3,i}\times X_i)+(A'_{3,j}\times X_j)\\ &\cdots\cdots\cdots\\ Y_{M-1}&=(A'_{M-1,i}\times X_i)+(A'_{M-1,j}\times X_j) \end{aligned} \qquad (11)$$

식(11)은 식(4)와 동일하다. 따라서 식(11)이 ***Lemma 3.2***를 만족하면 Y_1-Y_{M-1}의 해밍 가중치의 최소값은 '$M-1$'이 된다. 한편 $Y_0 = 0$이므로 $\boldsymbol{Y}$의 해밍 가중치의 최소값은 '$W_h(\boldsymbol{Y}) = M-2$'가 된다.

식(9)에서 $Y_1 = 0$이라고 가정하고 식(9)가 ***Lemma 3.2***를 만족하면 $Y_2 - Y_{M-1}$의 해밍 가중치의 최소값은 '$M-1$'이 된다. 앞에서 $Y_0 = Y_1 = 0$이면서 또 다른 $Y_i = 0$인 경우가 되는 경우는 없음을 확인하였으므로, $Y_0 \neq 0$, $Y_1 = 0$이므로 $\boldsymbol{Y}$의 해밍 가중치의 최소값은 '$W_h(\boldsymbol{Y}) = M-2$'가 된다.

이와 같은 과정을 $Y_i(3 \leq i \leq M-3)$에 대하여 반복하여 모두 ***Lemma 3.2***를 만족하면 Y의 해밍 가중치 최소값은 '$W_h(\boldsymbol{Y}) = M-2$'이 된다. □

Lemma 3.4

입력코드 $\boldsymbol{X}$의 해밍 가중치 '$W_h(\boldsymbol{X}) = m$ $(4 \leq m \leq M)$'이면, 출력코드 $\boldsymbol{Y}$의 해밍 가중치 최소값이 '$W_h(\boldsymbol{Y}) = M - m + 1$'이면 MDS 코드를 생성한다. 입력코드 $\boldsymbol{X}$의 원소 가운데 m 개의 원소가 '0'이 아니라면, 출력코드 $\boldsymbol{Y}$의 원소 $Y_p(0 \leq p \leq M-m)$를 '0'으로 가정하여 $\boldsymbol{X}$의 원소 가운데 하나를 다른 $\boldsymbol{X}$의 원소로 표현하고 이것을 $Y_q(p < q \leq M-m+1)$에 대입하여 $\boldsymbol{X}$의 원소 하나를 소거한다. Y_q는 '$W_h(\boldsymbol{X}) = m-1$'인 $M-q$ 차원의 행렬이 된다. 이러한 과정을 재귀적으로 반복하여 선형변환행렬이 MDS 코드를 생성하는지 여부를 판단한다.

증명

Lemma 3.3의 증명을 확장해서 재귀적으로 적용하여 증명할 수 있다. □

4. 구현 및 비교 검토

MDS 코드 생성여부를 판단하는 알고리즘에서 수행 시간이 많이 소요되는 연산은 곱셈 연산과 역수 연산이다. 본 논문에서 제안한 알고리즘과 기존 알고리즘에서의 곱셈 및 역수 연산 회수를 비교한다.

기존 알고리즘으로 선형변환행렬의 모든 정방부분행렬이 정칙이면 MDS 코드를 생성한다는 것이 연구되었다[10]. 행렬이 정칙이기 위한 필요조건은 행렬의 determinant(det)가 '0'이 아니어야 한다. 본 논문에서는 행렬을 상삼각행렬로 변환하고, 그 대각선 원소들이 모두 '0'이 아니면 det 또한 '0'이 아니라는 점을 이용하여 정칙행렬을 판단하였다.

이 알고리즘을 C언어를 사용하여 프로그램하고 수행하였다. 프로그램을 수행하여 곱셈 연산과 역수 연산이 얼마나 수행되는지 알아보았다. 선형변환행렬 크기에 따른 연산 수를 표 2에 나타내었다.

표 2 기존 알고리즘의 연산 수

	8×8	12×12	16×16	24×24	32×32
곱셈 연산	3.12×10^{5}	2.15×10^{8}	1.11×10^{11}	1.96×10^{16}	2.61×10^{21}
역수 연산	3.86×10^{5}	1.35×10^{7}	4.21×10^{9}	3.55×10^{14}	2.75×10^{19}

본 논문에서 제안하는 알고리즘은 반복적 연산을 필요로 하기 때문에 재귀호출 프로그램으로 작성하였고, C언어를 사용하여 구현하였다. 재귀호출 부분을 C로 구현한 것을 표 3에 나타내었다.

표 3 재귀호출 부분

```
int reduce(word32 va[M][M], int scnt, int cnt)

word32 vb[M][M], ka, kb ;
int ia, ib, ic ;

  if (cnt==1)
    for (ia=scnt ; ia<(M-1) ; ia++)
      if ( va[ia][1]==0 ) return 1 ;
 ka = mul(rev(va[ia][1], PRIM), va[ia][0], PRIM) ;
 for (ib=ia+1 ; ib<M ; ib++)
   if (mul(ka, va[ib][1], PRIM)==va[ib][0]) return 1 ;

    else
for (ia=scnt ; ia<(M-cnt) ; ia++)
   for (ib=0 ; ib<=cnt ; ib++)
      for (ic=ia ; ic<M ; ic++)
         vb[ic][ib] = va[ic][ib] ;
    if (vb[ia][cnt]==0) return 1 ;
    ka = rev(vb[ia][cnt], PRIM) ;
    for (ic=0 ; ic<cnt ; ic++)
       vb[ia][ic] = mul(ka, vb[ia][ic], PRIM) ;
    for (ib=ia+1 ; ib<M ; ib++)
      for (ic=0 ; ic<cnt ; ic++)
    vb[ib][ic] ^= mul(vb[ia][ic], vb[ib][cnt], PRIM) ;
    if (reduce(vb, ia+1, cnt-1)==1) return 1 ;

  return 0 ;
```

본 논문에서 제안하는 알고리즘도 프로그램을 수행하여 곱셈 연산과 역수 연산이 얼마나 수행되는지 알아보았다. 그리고 구한 연산 수를 표 4에 나타내었다.

표 4 변수를 소거하는 알고리즘의 연산 수

	8×8	12×12	16×16	24×24	32×32
곱셈 연산	7.75×10^{4}	2.03×10^{7}	4.87×10^{9}	2.79×10^{14}	1.64×10^{19}
역수 연산	1.12×10^{4}	2.49×10^{6}	5.66×10^{8}	3.10×10^{13}	1.78×10^{18}

두 알고리즘을 수행하여 구한 연산 수를 그림 11에 그래프로 나타내었다.

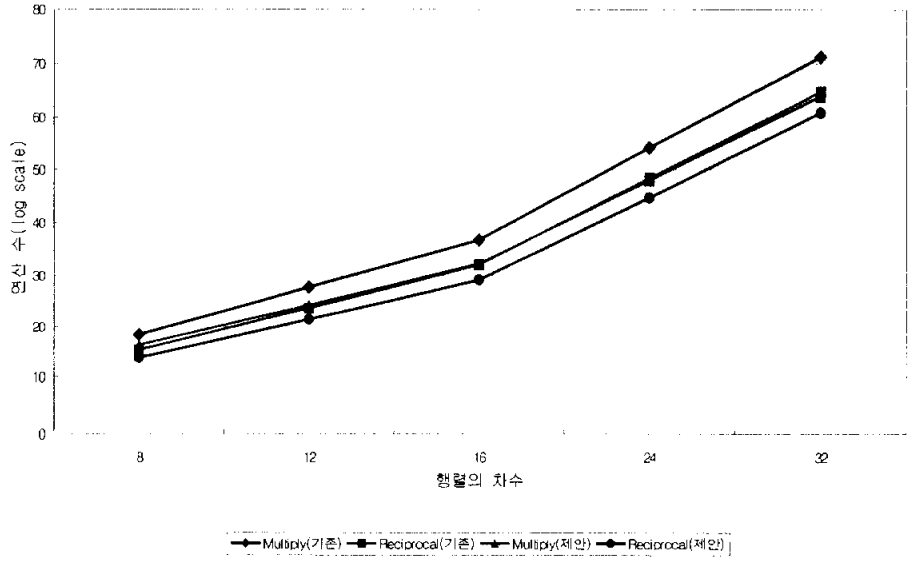


그림 11 연산 수 표시 그래프

두 알고리즘의 연산 수를 비교하여 보면 행렬의 차수가 커질수록 본 논문

에서 제안한 알고리즘의 연산 수가 큰 폭으로 감소하는 것을 알 수 있다. 그 결과 연산 수행 속도가 빨라져서 연산 수행 시간을 많이 단축할 수 있다.

5. 결 론

현재 많이 사용되고 있는 알고리즘은 블록 암호 알고리즘인데, 이 알고리즘에 대한 공격으로 선형 공격과 차분 공격이 있다. 블록 암호 알고리즘은 치환 계층과 교환 계층을 가지는데, 이들 공격에 강한 특성을 지니기 위해서 교환 계층에 선형변환행렬을 이용한다. 이 때 선형변환행렬이 MDS 코드를 생성하면 선형과 차분 공격에 강한 특성을 가진다.

본 논문에서는 블록 암호 알고리즘의 교환 계층의 선형변환행렬이 MDS 코드를 생성하는지 판단하는 알고리즘을 제안한다. 선형변환행렬의 입력 코드는 $GF(2^n)$상의 원소들로 구성되는데, 이 원소들을 변수로 해석하여 소거시키면서 연산을 수행하여 선형변환행렬이 MDS 코드를 생성하는지 확인한다. 본 논문에서 제안하는 알고리즘은 종래의 선형변환행렬의 모든 정방부분행렬이 정칙인지를 확인하는 알고리즘과 비교하여 연산 수를 크게 감소시켰다. 그 결과 연산의 수행 시간을 많이 단축시켰다.

향후 안전성이 강화된 블록 암호알고리즘을 개발하는데 본 논문에서 제안한 알고리즘이 많이 활용될 수 있을 것으로 기대가 된다.

참고문헌

[1] 한국정보보호센터, "128 비트 블록 암호 알고리즘(SEED) 개발 및 분석보고서", Dec. 1998.

[2] E. Biham and A. Shamir, "Differential cryptanalysis of DES-like crytosystems, Journal of Cryptology", vol. 4, no. 1, pp. 3-72, 1991.

[3] M. Matsui, "The first experimental cryptanalysis of the Data Encryption Standard, Advances in Cryptology", Proc. Of EUROCRYPT '91, Springer-Verlag, Berlin, pp. 1-11, 1994.

[4] H.M. Heys and S.E. Tavares, "The design of substitution-permutation networks resistant to differential and linear cryptanalysis", Proceedings of 2nd ACM Conference on Computer and Communications Security, Fairfax, Virginia, pp. 148-155, 1994.

[5] H.M. Heys and S.E. Tavares, "The design of product ciphers resistant to differential and linear cryptanalysis", Journal of Cryptology, Vol. 9, no. 1, pp. 1-19, 1996.

[6] F.J. MacWilliams and N.J.A. Sloane, "The theory of error correcting codes", North-Holland Publishing Company, 1977.

[7] S. Vaudenay, "On the need for multipermutations: Crytanalysis of MD4 and SAFER", Proc. of Fast Software Encryption (2), LNCS 1008, Springer-Verlag, pp. 286-297, 1995.

[8] V. Rijmen, J. Daemen, B. Preneel, A. Bosselaers, and E. De Win, "The cipher SHARK", Fast Software Encryption, LNCS 1039, D. Gollmann, Ed., Springer-Verlag, pp. 99-112, 1996.

[9] J. Daemen, L. Knudsen, and V. Rijmen, "The block cipher SQUARE", Proc. of Fast Software Encryption (4), LNCS, Springer-Verlag, 1997.

[10] A.M. Youssef, S. Mister, S.E. Tavares, "On the Design of Linear Transformation for Substitution Permutation Encryption Networks", in the Workshop Record of the Workshop on Selected Areas in Cryptography (SAC '97), pp. 40-48, Aug. 11-12, 1997.

[11] http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[12] Ju-Sung Kang, Choonsik Park, Sangjin Lee, and Jong-In Lim, "On the Optimal Diffusion Layers with Practical Security against Differential and Linear Cryptanalysis", Proceedings of ICISC'99, LNCS 1787, Spring-Verlag pp. 33-52, 1999.

[13] 박창수, 조경연, 송홍복, "SEED 형식 암호에서 공격에 강한 S 박스와 G 함수의 실험적 설계", 한국해양정보통신학회, TBD

[14] S. Miyaguchi, "The FEAL Cipher Family," Advances in Cryptology-CRYPTO '90 Proceedings, Springer-Verlag, 1991, pp.627-638.

[15] S. Murphy, "The Cryptanalysis of FEAL-4 with 20 Chosen Plaintexts," Journal of Cryptology, v.2,n.3, 1990, pp. 145-154.

[16] National Bureau of Standards, NBS FIPS PUB 46, "Data Encryption Standard", National Bureau of Standards, U.S. Department of Commerce, Jan 1997.

[17] X. Lai, J. Massey, and S. Murphy, "Markov Chiper and Differential Cryptanalysis", Advances in Cryptology-EUROCRYPT '91 Proceedings, Springer-Verlag, 1991, pp. 17-38.

[18] Nyberg and L.R. Knudsen, "Provable Security against Differential Cryptanalysis," Advances in Cryptology-CRYPTO '92, Proceedings, Springer-Verlag, 1993, pp566-574.

[19] A.Shimizu and S. Miyaguchi, "Fast Data Encipherment Algorithm

FEAL," Transactions of IEICE of japan, v.J70-D,n,7,Jul87, pp.1413-1423.

[20] M. Matsui and A. Yamagishi, "A New Method for Known Plaintest Attack of FEAL Cipher," Advances in Cryptology-EUROCRYPT '92 Proceedings, Springer-Verlag, 1993, pp.81-91.

[21] S.K. Langford and M.E. Hellman, "Cryptanalysis of DES," presented at 1994 RSA Data Security conference, Redwood Shores, CA, 12-14 Jan 1994.

[22] B.S. Kaliski and M.J.B Robshaw, "Linear Cryptanalysis Using Multiple Approximations," Advances in Cryptology-CRYPTO '94 Proceedings, Springer-Verlag, 1994, pp. 26-39.

감사의 글

오늘 이글을 남길 수 있도록 힘이 되어 주신 분들께 진심으로 감사드립니다.

2년 6개월의 대학 생활 동안 언제나 한결같은 따뜻한 보살핌과 밝은 웃음로 지도편달 해주신 지도교수님 조경연 교수님께 감사드립니다.

많이 부족한 저에게 관심을 가지시고 논문을 심사해주신 신봉기 교수님, 권오흠 교수님께 감사드립니다.

새벽녘까지 책을 들춰봐도 모르는 것들이 너무나 많았지만, 그때 마다 힘이 되어 주시고 이 글을 쓰는 동안 많은 도움을 주신 박창수 선배님께 감사드립니다.

함께 공부하며 웃음을 나누었던 김길호 선배님과 동기분들 그리고 마이크로 프로세스 연구실의 손창우, 한경현, 전희성, 장현지, 조민경에게 감사드립니다.

바쁜 업무 중에도 무사히 학업을 마칠 수 있도록 도와주신 남양유업(주) 부산지점 선후배님들께도 감사드립니다.

나름대로 최선을 다한 2년 6개월 이었다고 생각했는데, 막상 지금에 와 보니 부족한 점이 너무나 많습니다. 전역 후 가진 첫 직장에서 더 나은 나의 발전을 위해 의욕적으로 시작한 공부였습니다. 아침부터 저녁까지 회사업무에 이은 학업으로 항상 지쳐 있었지만 배움이 있어 즐거웠습니다. 누군가가 내일을 대신하고 있지는 않을까 하는 미안함과 걱정도 있었지만, 많은 분들의 도움으로 무사히 학업을 마칠 수 있어 기쁘고, 부족하나마 지금껏 배운 것을 소중히 여기고 활용하겠습니다.

끝으로, 묵묵히 아들을 믿고 사랑으로 지켜봐 주시고 돌보아 주신 아버지, 어머니와 언제나 동일하신 하나님께 감사드립니다.

2003년 8월

윤 성 훈 올림