소수 코드를 이용한 블록 암호화 알고리즘

송 문 빈, *오 재 곤, 정 연 모
경희대학교 전자공학과,
*한국산업기술대학교 전자공학과

A BLOCK CRYPTOGRAPHIC ALGORITHM BASED ON A PRIME CODE

MoonVin Song, *Chae-Gon Oh, Yunmo Chung
Dept. of Electronics Engineering, Kyung Hee University
Dept. of Electronics Engineering, Korea Polytechnic University
chung@nms.kyunghee.ac.kr

Abstract

In this paper, we propose a prime code and a new cryptographic algorithm for encryption and decryption as its application. The characteristics of prime numbers with irregular distribution and uniqueness are used to generate the prime code. Based on the prime code, an encryption algorithm for secret key is presented. Since the algorithm requires simpler operations than existing encryption such as DES, the burden for hardware implementation of the encryption and decryption process is alleviated.

I. 서론

현대사회는 첨단 정보통신 및 정보처리를 기반으로 하는 정보화 사회이다. 따라서 공간적, 시간적 제약을 뛰어넘어 범세계적인 규모의 네트워크화로 발전되어 가고 있다. 전자 우편과 전자 상거래가 널리 이용됨에 따라 인터넷을 통한 개인 정보 교환이 빠르게 확산되고 이에 따른 부작용 또한 급속도로 확산되고 있으므로 정보보호의 문제가 대두되고 있다. [1]

이와 같이 전자적으로 교환되는 문서들의 보안(security) 문제를 해결하려면 정보 보호의 핵심이 되는 암호화가 필수적으로 선행되어야 한다. [3]

본 논문에서는 소수(prime number)의 불규칙 분포성과 유일성을 이용해 소수 코드를 생성하고 이 소수 코드를 사용해 비밀키 암호화를 위한 새로운 암호화 알고리즘을 제시한다. 이 알고리즘은 기존의 DES 및 SEED 알고리즘에 비해 상대적으로 간단해진 연산과 소수 코드로의 변환을 이용하므로 암호화에 대한 하드웨어적인 부담을 줄였다.

II. 소수 코드의 생성

소수는 1과 자신만을 소인수로 가지는 자연수를 말하며 자연수체계 내에서 소수의 분포는 불규칙성을 가지기 때문에 소수를 수학적 식으로 표현할 수 없다. 소수 코드를 사용하면 서로 다른 소수의 합으로 모든 자연수를 표현할 수 있다. 이러한 특성들을 사용하여 소수들로 구성된 소수 코드(prime code)를 생성하고 이를 암호화(encryption) 및 복호화(decryption)에 이용한다.

소수 코드는 앞에서 언급한 소수의 성질을 이용하여 생성한다. 먼저 생성하고자 하는 n비트(bit) 각각의 가중치를 설정하고 설정된 각 비트의 가중치를 이용하여 생성하며 생성 단계는 다음과 같이 4단계로 나눌 수 있다.

단계1 [기본 비트] : 소수 코드 각 비트의 가중치를 소수로 정의한다. 첫 번째 비트(즉 비트1)의 가중치를 1로 한다.

단계2 [소수 코드 체계] : (n-1)번째 비트까지의 가중치들의 합을 K라고 가정할 경우에 K+1이 소수이면 n번째 비트의 가중치를 K+1로 설정한다. 또한 K+1이 소수가 아닌 경우에 K가 소수이면 이를 n번째 비트의

가중치로 설정하고, 그렇지 않으면 K보다 작은 가장 근접한 소수를 n번째 비트의 가중치로 한다. 8비트의 소수 코드를 위해서 단계1과 단계2의 과정을 거치면 <표 1>과 같이 각 비트의 가중치를 설정할 수 있다.

<표 1> 이진코드와 소수코드의 가중치 비교

	이 진 코 드								소 수 코 드							
비 트	8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1
가중치	128	64	32	16	8	4	2	1	97	47	23	13	7	3	2	1

단계3 [소수 코드 생성 테이블] : 자연수를 소수 코드로 변환하기 위해 사용된 소수 코드 생성 테이블(prime code generation table)의 구성에 대해서 설명하기로 한다. 자연수 M을 소수 코드 체계의 각 비트의 가중치들과 비교하여 M과 같거나 작은 가장 근접한 테이블의 비트 i의 해당 자연수에 M을 써준다. 여기서 비트 i 의 가중치를 R이라고 하고, 이 경우에 (M-R)이 0이면 수행을 종료한다. 만일 0이 아니면 (M - 지금까지 고려된 가중치의 합) M과 R의 차에 대해서 위의 과정을 반복한다. 단 여기서도 테이블에는 해당 비트에 M을 써준다.

자연수 11의 경우를 고려해 보자. 같거나 가장 근접한 작은 소수 코드의 가중치는 7 이므로 해당하는 비트 4에 11을 적어 준다. 계속해서 11과 가중치 7과의 차인 4와 같거나 가장 근접한 작은 가중치는 3이므로 이에 해당하는 비트 3에 11을 적어주고, 11과 가중치의 합 10(=7+3)과의 차 1은 가중치 1과 같으므로 비트 1에 11을 적어 주면, 자연수 11에 대한 소수 코드 생성 테이블은 완성이 된다.

이와 같은 작업을 해당 자연수에 대해서 수행하면 <표 2>와 같은 소수 코드 생성 테이블을 만들 수 있다.

단계4 [소수 코드 생성] : 소수 코드 생성 테이블을 이용하여 십진 코드를 소수 코드로 변환할 수 있다. 변환하고자 하는 자연수가 들어있는 비트들을 1로 하면 소수 코드를 만들 수 있다. 예를 들어 자연수 69를 고려해 보자 소수 코드 테이블에서 69가 들어있는 비트인 비트 7, 5, 4, 2를 1로 하면 소수 코드로 69는 01011010을 만들 수 있다.

소수코드를 십진 코드로 변환 시에는 소수 코드 생성 테이블에서 소수 코드의 비트 중에서 1에 해당하는 비트의 가중치만을 더하면 된다. 예를 들면 소수코드 01011010을 십진 코드로 변환하려면 1이 있는 디지트의 가중치인 47, 13, 7, 2를 더해주면 69로 변환이 된다.

소수코드의 가중치는 <표1> 과 같이 불규칙하므로 소수 코드의 수가 증가 시는 불규칙 적으로 증가한다. 이는 소수코드 생성 테이블을 이용하지 않고서는 소수 코드의 예측을 불가능하게 한다. 이를 암호화에 이용하면 입력 데이터의 정보를 모두 가지면서 입력 데이터와는 모양이 전혀 다른 새로운 데이터로 변환을 할 수 있다. 8 비트의 이진코드를 소수코드로 변환을 하면 9비트로 확장되는데 이를 암호화에 이용하면 이진코드가 소수코드로 변환되면서 비트의 수가 증가하므로 변환된 데이터에서 원래의 데이터를 유추해 낼 수

<표 2> 소수코드 생성 테이블

비 트	8	7	6	5
가 중 치	97	47	23	13
해당자연수	97~127 ...	47~96 ...	3~46, 70~96 120~127 ,,,	13~22, 36~46 60~69, 83~96 110~119 ...

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

있는 확률을 떨어트린다.

Ⅲ. 소수코드를 이용한 암호화 과정

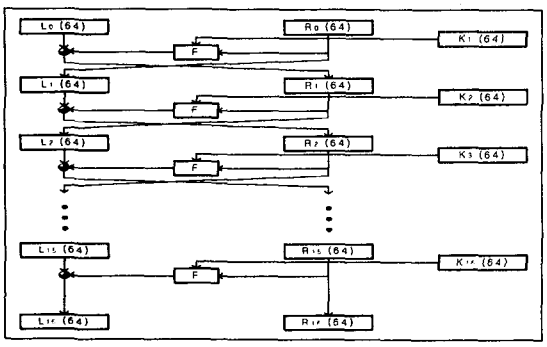
소수 코드를 이용한 암호화 과정은 128비트의 입력 데이터와 128비트의 키를 가지고 암호화를 한다. 비밀키 기반 중에서 블록 암호화 방식의 알고리즘의 안정성은 키의 길이에 기반하고 있으며, 현재 128비트의 키를 권고하고 있다.

키의 길이를 길게 하는 것이 안정성을 높일 수 있는 하나의 방법이다. 그러나 하드웨어와 소프트웨어적으로 실제 구현을 하였을 때의 trade-off를 고려해 소수 코드를 이용한 암호화 알고리즘은 128비트의 키를 가지고 암호화를 한다. [7]

알고리즘 구상 과정부터 하드웨어적인 설계를 고려하여 하드웨어로 구현이 쉬운 비트 연산 및 비트의 자리바꿈을 많이 사용하여 알고리즘을 설계하였다. 기존의 블록암호화 방식의 알고리즘인 DES 및 SEED를 하드웨어로 설계할 때 S박스의 변환되는 값들을 저장하기 위한 ROM을 사용해야 한다. 그리고 그 ROM을 제어하기 위한 제어 회로가 추가되어야만 한다. 그러나 본 암호화 알고리즘에서는 소수 코드로의 변환을 이용하여 F 함수를 구현한다. 소수 코드로의 변환은 S박스와 기능적인 측면에서 같은 일을 하면서도 ROM을 사용할 필요가 없다. 기존의 DES 및 SEED와 같은 블록 암호화 알고리즘이 가지고 있는 S박스를 소수코드의 변환이 그 기능을 대신하고 있다. 따라서 메모리를 사용하지 않으며, 소수 코드로 변환하는 부분을 제외한 모든 부분이 xor 연산과 자리바꿈 연산으로만 이루어지므로 계산이 간편하다.

따라서 하드웨어 설계시에 기존의 DES 및 SEED 블록 암호화 알고리즘에 비해 게이트의 숫자에서의 감소와 함께 수행 속도의 향상을 기해 할 수 있다.

<그림 1>은 소수코드를 이용한 암호화 알고리즘을 나타낸 블록도 이다.

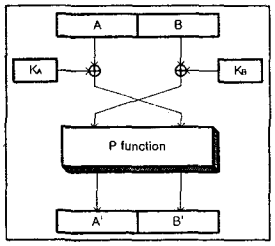


<그림 1> 소수 코드를 이용한 암호화 알고리즘

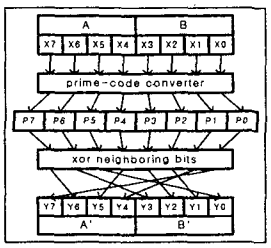
입력 데이터 128비트를 각 64비트씩 L_i과 R_i로 나눈다. 128비트로 입력된 킷값을 각 라운드에 해당하는 64비트의 라운드 키를 생성한다. 생성된 라운드 키 K_i와 R_i를 F함수의 입력으로 사용하여 수행한 결과를 L_i과 xor 연산한 후에 이를 R_{i+1}의 입력으로 하고, Ri는 L_{i+1}의 입력으로 한다. 이와 같은 과정을 16라운드 거친 L_{16}과 R_{16}를 최종 출력으로 하여 암호화된 결과를 얻는다. 단순한 구조를 여러 번 반복하여 사용하면 복잡도가 증가한다는 Luby와 Rackoff의 이론에 근거하여 같은 구조로 이루어진 각 라운드를 16 라운드 반복한다.

F함수는 8 비트의 이진코드를 9 비트의 소수코드로 확장 변환을 하는 P블록을 포함하고 있으며, 라운드 키 생성 블록과 함께 블록 암호화 알고리즘을 이루는 중요한 부분이다.

<그림 2>는 F 함수의 블록도 이고 <그림 3>은 P 함수의 블록도 이다.



<그림 2> F 함수

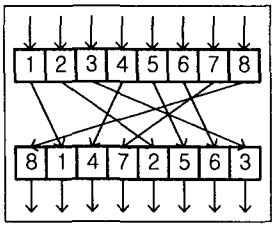


<그림 3> P 함수

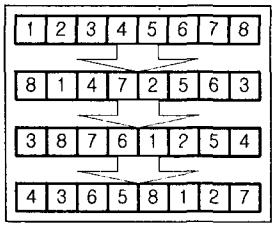
<그림 2>의 F 함수에서 입력 데이터인 64비트의 R_i를 좌우 32비트씩 각 A와 B로 나눈다. 64 비트인 라운드 키 K_i 도 각 32비트인 K_A 와 K_B로 나눈다. 각 32 비트의 A 와 K_A, 그리고 B 와 K_B를 각각 xor 연산한 후에 P 함수의 입력으로 사용한다. 32비트인 A′와 B′는 P 함수의 연산 결과로서 F 함수의 최종 출력이다.

<그림 3>의 P 함수는 이진 코드를 소수 코드로 변환을 하는 과정을 거쳐 F 함수의 기본적인 연산을 수행하는 블록이다. 즉 각 32비트인 A와 B를 입력으로 받아서 8개의 8비트 블록 X_i로 나눈다. 나누어진 8비트의 이진 코드 값을 가진 X_i 블록은 각각 소수 코드로 변환을 하는 소수 코드 변환기(prime code converter)를 거친다. 소수 코드 변환기의 출력은 확장된 8개의 9비트의 P_i블록으로서 총 72비트의 크기를 갖는다. 9비트인 각 P_i블록 내부의 비트들은 각각 이웃하는 비트들간의 xor 연산을 거쳐 각 8비트인 Y_i 블록으로 변환된다. 각 Yi 블록들을 <그림 4>에서와 같이 일정한 순서로 자리를 바꾸어 4 블록씩 합쳐서 A′와 B′를 생성한다.

<그림 5>는 여러 라운드를 거쳐서 블록별로 자리를 바꾼 결과를 나타내었다.



<그림 4> 블록별 자리바꿈



<그림 5> 자리바꿈 결과

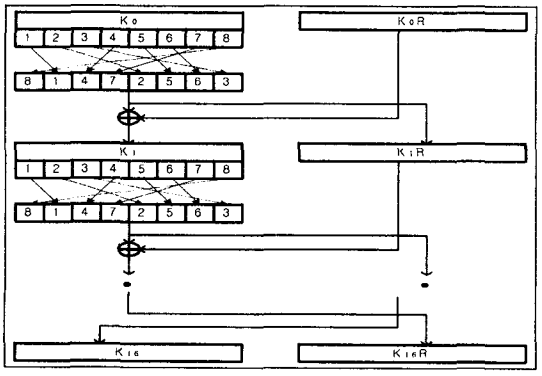
<그림 5>에서와 같이 자리바꿈을 반복해도 같은 위치에 동일한 블록이 오지 않음을 확인 할 수 있다. 이렇게 변환된 블록은 혼돈을 증가시킬 수 있다.

소수 코드 변환블록은 소수 코드 생성 알고리즘에 의해 8비트의 2진 코드를 9비트의 소수 코드로 변환하는 블록이다. P 함수에서는 9 비트로 변환된 소수 코드는 데이터의 위치에 혼돈을 주어 복잡성을 증가시키기 위하여 최종 출력은 <표 3>과 같이 각 비트의 위치를 바꾸어 출력한다.

<표 3> 변환 소수 코드의 위치 변환

소수 코드	193	97	47	23	13	7	3	2	1
변 환 소수 코드	3	293	7	1	47	2	13	97	23

블록 암호화 알고리즘에서 키 생성 블록은 F 함수와 함께 중요한 블록이다. 각 라운드마다 적용되는 키는 각기 다른 값을 가지고 있어서 유추하기 어려워야 한다. 따라서 본 논문에서는 <그림 6>에서와 같이 블록별 자리바꿈과 xor 연산을 적용하여 라운드 키를 생성한다.



<그림 6> 라운드 키 생성

입력된 128비트의 키는 라운드 키 생성 알고리즘에 의해서 각 라운드를 위해서 64비트의 라운드 키 K_i를 생성한다. 128비트의 입력키를 각 64비트인 K_i와 K_iR로 나눈다. 64비트의 K_i를 8비트인 8개의 블록으로 나누어 <그림 4>와 같이 동일한 방법으로 블록별 자리바꿈을 수행하여 복잡도를 증가시킨다. 자리바꿈한 결과는 $K_{i+1}R$의 입력으로 사용하고, K_iR과 xor 연산을 수행하여 라운드 키 K_{i+1}의 입력으로 사용한다. 이렇게 동일한 과정을 16번 반복하여 16개의 라운드 키를 생성한다.

Ⅳ. 결과 및 결론

소수 코드는 정수 내에서 소수가 가지는 특징인 유일성 및 불규칙 분포 성을 바탕으로 정의된 코드이다. 본 논문에서는 이러한 소수 코드의 특징을 암호화 알고리즘에 적용하여 소수 코드를 이용한 암호화 알고리즘을 정의하였다. 소수 코드를 이용한 암호화 알고리즘은 DES 및 SEED 알고리즘과 같이 S박스를 사용하지 않고 소수 코드 변환을 적용하였다. 따라서 본 알고리즘은 소수 코드로의 변환과 블록별 자리바꿈 및 xor연산을 주로 사용하여 DES 및 SEED 알고리즘에 비해 수식적인 연산과정이 없으므로 간단하고, 소프트웨어 및 하드웨어로의 구현이 용이다. [3][4]

참고문헌

[1] 류승석 · 오재곤 · 정연모, "GOST 알고리즘을 이용한 암호화 칩 설계에 관한 연구", 대한전자공학회 CAD 및 VLSI 설계 연구회 학술발표회 논문집, PP. 49-54, 1997.

[2] Feistel, "Cryptography and Computer Privacy", Scientific American, May 1973.

[3] "Data Encryption Standard," National Bureau of Standard, Federal Information Processing Standards Publication 46, Jan. 1977.

[4] T. A. Berson, "Long key variants of DES," proc. crypto'82.

[5] D. R. Stinson, "Cryptography", CRC Press, 1995.

[6] 조주연, 이필중, "Differential Cryptanalysis에 관한 고찰," 한국통신정보보호학회 학회지 제 3권 1호, 1993.

[7] 류승석, "암호화 칩의 설계에 관한 연구", 경희대학교 석사학위 논문집, 1988.

[8] Bruce Schneider, "Applied Cryptography", John Wiley & Sons, 1996.