□研究論文□

韓國 軍事運營分析 學會誌
第15卷, 第2號, 1989. 12. 31.

원시 이원 BCH 부호를 사용한 Algebraic-Coded Cryptosystem (Algebraic-Coded Cryptosystem Using Primitive Binary BCH Codes)

조 용 건* 남 길 현**

Abstract

The concept of Algebraic-Coded Cryptosystem has been proposed recently but its application has not been developed yet. The primary object of this paper is to implement the Private-Key Algebraic-Coded Cryptosystem by using the primitive binary BCH codes. In the analysis of the cryptosystem, we find out the fact that there may exist other key pairs S_i and P_i satisfying $G^*=S_iG_sP_i$ where SG_sP is the original cryptosystem made by use of the systematic code generation matrix G_s.

1. 서 론

암호통신을 하는 것은 더이상 특수한 정부 조직에서만의 일이 아니라 컴퓨터 통신을 하는 모든 집단에 필요하게 되었고 많은 응용 프로그램들이 개발되었다.

본 연구의 목적은 지금까지의 통상적인 암호화 방법인 재배열 방법(transposition cipher), 치환방법(substitution cipher) 또는 이들을 복잡하게 반복 이용하는 혼합 방법과는 성격이 다른 large distance algebraic 부호를 사용하는 McEliece Public-key Crypto-

* 육군중앙전산소
** 국방대학원

system (MPBC) [5]과 비교적 간단한 코드를 사용하는 private-key 크립토시스템인 Private-key Algebraic-coded Cryptosystem (PRAC) [6]을 분석하여 PRAC을 원시 이원(primitive binary) BCH 부호로서 구현하고 크립터널리스트의 입장에서 MPBC와 PRAC의 크립토컴플렉시티를 분석하는데 있다.

G의 행은 (n, k) 선형코드 C를 생성(또는 span)하기 때문에 행열 G는 C의 생성행열이라 한다. 어떠한 k×n 생성행열 G에 대해서도 n-k 개의 선형독립인 행을 가지고 있는 (n-k)×n 행열 H가 존재하여 G의 row space에 있는 어떠한 vector도 H의 행에 대해서 orthogonal이고 H의 행에 대해서 orthogonal인 어떠한 벡터도 G의 row space 내에 있다. 따라서 G에 의해 생성되는 임의의 tuple v는 $v * H^T = 0$ 이기만 하면 G에 의해 생성되는 부호어가 된다. 그리고 (n, k) 선형코드 C 내의 임의의 부호어를 i 번 순회 치환시킨 부호어가 역시 C내의 부호어가 될 때 순회부호라 한다. [4]

2.5. BCH 부호 (Bose-Chaudhuri-Hocquenghem Codes)

BCH 부호는 순회부호의 일종이며 강력한 다중 오류정정능력을 갖는 부호이다. BCH 부호는 단일 오류만을 정정할 수 있는 Hamming 부호를 여러 개의 산발오류를 정정할 수 있는 다중 오류정정부호로 확대시킨 것이므로 Hamming 순회부호는 BCH 부호의 가장 단순한 경우라 할 수 있다.

어떠한 자연수 m (m〉=3)과 t에 대해서도 다음과 같은 parameter를 갖는 이원 BCH부호가 존재한다.

블록길이 (Block length) : $n = 2^m - 1$

정보 digit 길이 : k

parity-check digit 의 수 : $n-k \leq mt$

최소거리 (Minimum distance) : $d_{min} \geq 2t+1$

Galois체 CF (2^m)의 원시원(primitive element) α로부터 얻어지는 이원 BCH부호를 생각해 보자. 길이 2^m-1인 t-error-correcting BCH 부호의 생성다항식 g(x)는 $\alpha, \alpha^2, \alpha^3, \cdots, \alpha^{2t}$를 근으로 갖는 GF(2)상의 최소 차수의 다항식이다. 따라서 g(x)는 그의 근 $\alpha, \alpha^2, \cdots, \alpha^{2t}$에 대응하는 최소다항식들의 최소 공배다항식이어야 한다. $\phi_i(x)$가 a^i의 최소다항식 이라면 g(x)는 다음과 같다.

$g(x) = LCM[\phi_1(x), \phi_2(x), \cdots, \phi_{2t}(x)]$.

만약 $v(x) = v_0 + v_1X + \cdots + v_{n-1}X^{n-1}$ 가 길이 $n = 2^m-1$인 t-error-correcting BCH 부호중 하나의 부호다항식이라 하자. α^i가 v(X)의 근이므로 $1 \leq i \leq 2t$에 대해서

$v(a^i) = v_0 + v_1a^i + v_2a^{2i} + \cdots v_{n-1}a^{(n-1)i} = 0$가 성립한다.

이식을 binary 연산자인 곱하기 "*"을 사용하여 행열의 형태로 표현하면 다음과 같다.

$$(v_0, v_1, \cdots, v_{n-1}) \quad * \quad \begin{bmatrix} 1 \\ \alpha^i \\ \alpha^{2i} \\ \cdot \\ \cdot \\ \alpha^{(n-1)i} \end{bmatrix} = 0, \quad \text{for} \quad 1 \le i \le 2t \qquad (2.1)$$

(2.1) 식은 $(v_0, v_1, \cdots, v_{n-1})$ 과 $(1, \alpha^i, \alpha^{2i}, \cdots, \alpha^{(n-1)i})$ 의 내적이 영(zero) 이라는 뜻이다. 따라서 다음과 같은 행열 H를 생각할 수 있다.

$$H = \begin{bmatrix} 1 & \alpha & \alpha^2 & \alpha^3 & \cdots & \alpha^{n-1} \\ 1 & (\alpha^2) & (\alpha^2)^2 & (\alpha^2)^3 & \cdots & (\alpha^2)^{n-1} \\ \cdot & \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & \cdot & & \cdot \\ 1 & (\alpha^{2t}) & (\alpha^{2t})^2 & (\alpha^{2t})^3 & \cdots & (\alpha^{2t})^{n-1} \end{bmatrix}$$

(2.1) 식으로부터 $v=(v_0, v_1, \cdots, v_{n-1})$ 이 t-error-correcting BCH 부호의 부호어이려면 $v*H^T=0$ 이어야 한다. 이때 H를 parity-check 행열이라고 한다. [4]

3. Algebraic-Coded Cryptosystem (ACC)

McEliece는 algebraic coding 이론에 근거하여 public-key 크립토시스템을 소개하였다 [5]. 이 크립토시스템은 t-error correcting Goppa 부호에 대응하는 생성행열을 사용하도록 설계되어 있으며 이원 BCH 부호와 Reed-Solomon 부호를 사용하여 구현할 수도 있다 [9]. 만약 MPBC에서의 public 키가 비밀이 유지된다면 간단한 부호로서도 상대적으로 높은 비도를 유지할 수 있는데 이 방법을 사용한 크립토시스템이 Private-key Algebraic-coded Cryptosystems (PRAC) 이다[6]. 본 논문에서는 MPBC와 PRAC 모두를 Algebraic-Coded Cryptosystems (ACC) 로 고려하였다.

3.1. McEliece Public-key Cryptosystem (MPBC)

가. 암호화 방법 (Encryption)

G를 GF(2) 상에서 t-error correcting을 할 수 있는 k×n 생성행열이라하자. 이때 부호의 정보 비율은 k/n이다. 임의의 n×n 순열행열 (Permutation Matrix) P와 스크램블러 (Scrambler) 라 불리는 k×k 비특이행열 S를 선택하여 $G^*=SGP$를 만든다. 이것으로 부터 암호화는 다음과 같이 표현할 수 있다.

$C=MG^*+Z$

단, C : 길이가 n 인 ciphertext,

M : 길이가 k 인 plaintext,

Z : Hamming weight $t'<=t$이고 길이가 n인 랜덤 에러벡터

나. 해독방법 (Decryption)

해독은 암호화 방법의 역순으로 하면 된다. 먼저 암호문

$C = MG^* + Z$로 부터

$G^* = SGP$

$C = MG^* + Z$

$= MSGP + Z$

$= M^*GP + Z$이 된다. (여기서 $M^* = MS$)

따라서 다음 단계를 거쳐서 M을 찾는다.

단계 1. C^* 를 계산한다.

$C^* = CP^T = M^*G + ZP^T$(여기서 P^T는 P의 전치 행열)

$= M^*G + Z$ (여기서 $Z^* = ZP^T$)

단계 2. error를 찾아내서 M^*를 찾아낸다.

Z^*를 신드롬(syndrome)으로 부터 찾아낸다.

$C^*H^T = M^*GH^T + Z^*H^T$

$= Z^*H^T$

올바르게 수정된 암호문으로 부터 M^*를 찾는다.

단계 3. M^*로부터 M를 찾는다.

$M = M^*S^{-1}$

이 시스템의 암호화 및 해독 알고리즘은 매우 단순하여 t를 크게 함으로써 비도를 증가시킬 수 있다. 즉 $n=1024$, $k=524$, $t=50$이라면 MPBC는 거의 10^{19}정도의 워크 팩터를 필요로 한다. 그러나 MPBC는 50개의 error를 수정해야 하기 때문에 암호 및 해독시 높은 overhead와 낮은 정보율(information rate)로 인하여 컴퓨터 통신에 효과적인 크립토시스템이라고 볼 수는 없다.

3.2. Private-key Algebraic-Coded Cryptosystem (PRAC)

MPBC의 암호 및 해독의 overhead를 줄이고 정보율을 높이기 위해 PRAC이 제안되었다[6, 9]. PRAC의 알고리즘은 MPBC와 같다. 그러나 PRAC은 S, G, P와 함께 G^*도 비밀 키로 유지하며 PRAC을 해독하기 위해서 크립터널리스트는 G^*를 먼저 알아내야 한다. 따라서 PRAC은 간단단 error-correcting-code로서도 더 높은 비도를 제공할 수 있다. PRAC은 ciphertext-only attack에는 높은 비도를 보장하지만 known-plaintext attack에서는 많은 양의 보통문과 암호문의 쌍을 필요로 하고 chosen-plaintext attack에서는 MPBC와 거의 대등하게 된다. [6]

3.3 (15, 7) BCH 부호 생성 행열

$m=4$일때 원시다항식 $p(X) = 1 + X + X^4$을 사용하면 Table 4와 같은 Galois체 $GF(2^4)$를 만들 수 있다. α, α^3의 최소다항식은

$\phi_1(X) = 1 + X + X^4$

$\phi_3(X) = 1 + X + X^2 + X^3 + X^4$이고

길이 $15(n=2^4-1)$인 2-error-correcting BCH 부호는 다음과 같은 생성 다항식에 의해 만들어진다.

$$g(x) = LCM[\phi_1(X), \phi_3(X)]$$

$\phi_1(X)$와 $\phi_3(X)$는 서로 다른 기약다항식(irreducible polynomial)이기 때문에 $g(X) = \phi_1(X) * \phi_3(X) = 1+X^4+X^6+X^7+X^8$이 된다.

이에 대응하는 행열 형태는 다음과 같다.

$$G = \begin{bmatrix} 100010111000000 \\ 010001011100000 \\ 001000101110000 \\ 000100010111000 \\ 000010001011100 \\ 000001000101110 \\ 000000100010111 \end{bmatrix}$$

TABLE 4 $p(X) = 1+X+X^4$에 의해 생성되는 $GF(2^4)$의 ELEMENTS의 표현

Power 표 현	다항식 표 현	4-Tuple 표 현
0	0	(0 0 0 0)
1	1	(1 0 0 0)
α	α	(0 1 0 0)
α^2	α^2	(0 0 1 0)
α^3	α^3	(0 0 0 1)
α^4	$1+\alpha$	(1 1 0 0)
α^5	$\alpha+\alpha^2$	(0 1 1 0)
α^6	$\alpha^2+\alpha^3$	(0 0 1 1)
α^7	$1+\alpha+\alpha^3$	(1 1 0 1)
α^8	$1+\alpha^2$	(1 0 1 0)
α^9	$\alpha+\alpha^3$	(0 1 0 1)
α^{10}	$1+\alpha+\alpha^2$	(1 1 1 0)
α^{11}	$\alpha+\alpha^2+\alpha^3$	(0 1 1 1)
α^{12}	$1+\alpha+\alpha^2+\alpha^3$	(1 1 1 1)
α^{13}	$1+\alpha^2+\alpha^3$	(1 0 1 1)
α^{14}	$1+\alpha^3$	(1 0 0 1)

이 G 행열을 elementary row operation을 수행하여 다음과 같은 조직형 생성행열로 변환이 가능하다. [10]

$$G = \begin{bmatrix} 100010111000000 \\ 110011100100000 \\ 011001110010000 \\ 101110000001000 \\ 010111000000100 \\ 001011100000010 \\ 000101110000001 \end{bmatrix}$$

암호화 및 복호화 과정은 PRAC과 동일하므로 생략한다.

4. MPBC와 PRAC의 크립토컴플렉시티 및 원시 이원(255, 179) BCH 부호

4.1 가정사항

가. 크립터널리스트는 크립토시스템의 암호 및 해독 알고리즘을 잘 알고 있다.

나. 크립터널리스트는 보통문과 암호문의 쌍(pair)을 충분히 갖고있다. 즉, 크립토시스템은 어떠한 known-plaintext attack 으로부터도 안전해야 한다. 또한 크립토시스템이 chosen-plaintext attack으로 부터 안전하면 이 크립토시스템은 가장 강한 시스템으로 간주된다.

다. DES는 현재 사용되고 있는 암호시스템이다. DES를 비밀 키의 반복적인 방법(exhaustive method)에 의해 깨뜨리기 위해서는

약 2^{56}DES 워크팩터를 필요로 한다. 비록 2^{56} $\approx 10^{17}$만큼의 워크팩터가 매우 크지 않다 해도 그것은 DES가 현재의 기술 여건하에서는 사용할 만한 비도 수준(acceptable level of security)를 제공한다고 본다. 그러므로, 만약 어떤 암호시스템의 크립토컴플렉시티가 DES 만큼의 order를 가진다면 그 크립토시스템은 안전하다고 간주한다.

라. 암호화하는 동안에 랜덤 에러벡터는 매우 중요한 역할을 가지며, PRAC의 비도는 결정적으로 에러벡터의 임의성(randomness)에 달려있다. 여기에서는 임의성이 충분한 랜덤 벡터를 생성해 주는 난수 발생기가 있다고 가정한다.

마. 언어 통계학적으로 모든 가능한 문자의 무작위 조합에서 의미있는 메시지의 퍼센트가 매우 적다는 것을 알 수 있다. 그래서 만약 의미있는 메시지를 어떤 크립터널리시스에 의해 주어진 암호문으로부터 얻는다면, 그때 그 메시지는 주어진 암호문에 대응하는 보통문이라고 간주한다. [6]

4.2 MPBC의 크립터널리스트 ATTACK

McEliece가 제시한 [5] 크립터널리스트가 취할 수 있는 두가지 방법이 있다.

가. G*로부터 S, G, P의 도출

G*로부터 정확한 S, G, P를 도출하기란 어렵다. 그러나 크립터널리스트의 입장에서는 S_i, G_i, $P_i = G^*$, G_i는 t-error correcting code를 만족하는 것만을 원한다.

G*를 알고 있는 크립터널리스트는 $G^* = S_j$, G_a, P_j, G_a는 조직생성행열을 만족하는 S_j, G_a, P_j를 elementary row operation과 column operatoion을 통해서 구할 수 있다. 이때 G*, G_a, G를 combinatorially equivalent하다고 한다. 만약 G의 최소 거리가 적은 부호라면 키는 비교적 쉽게 구할수 있다. 그러나 G가 large distance Goppa 부호라면 위의 방법으로 키를 얻기 위해서는 매우 큰 워크 팩터가 필요하다.

나. 키 없이 직접 C로 부터 M을 복구

M과 C의 pair로 부터 n개의 방정식을 만들어 Z를 고려해서 M을 찾아낸다. M과 C를 [M, C] pair(보통문-암호문 짝)이라 하고, Z를 랜덤 에러벡터라하자. 그러면,

$M = m_1 m_2 \cdots m_k$,

$C = c_1 c_2 \cdots c_k \cdots c_n$,

$Z = z_1 z_2 \cdots z_k \cdots z_n$,

$G^* = [G_{ij}^*]$, for $i = 1, \cdots, k$, and $j = 1, \cdots, n$

t-error correcting algebraic code.

MPBC의 암호화 방법으로부터 보통문을 구하기 위하여 다음 방정식을 만든다.

$$c_1 = m_1 G_{11} + m_2 G_{21} + \cdots + m_k G_{k1} + z_1$$
$$c_2 = m_1 G_{12} + m_2 G_{22} + \cdots + m_k G_{k2} + z_2$$
$$\cdots \quad \cdots \quad \cdots \quad \cdots \quad \cdots \quad \cdots$$
$$c_n = m_1 G_{1n} + m_2 G_{2n} + \cdots + m_k G_{kn} + z_n$$

k개의 미지수($m_1, m_2, \cdots, m_k$)를 구하려면 k^3 오퍼레이션을 해야 한다. t의 갯수가 n-k보다 적기때문에 크립터널리스트는 n개의 방정식에서 error를 포함하지 않는 k개의 식을 선택할 확률이 존재한다. 따라서 크립터널리스트는 의미있는 메시지를 얻을 때까지 n개의 식에서 임의의 k개식을 선택하여 계속적으로 방정식을 풀려고 할 것이다. k개의 방정식에서 error가 없을 확률 P_k는 $\prod_{i=0}^{k-1}(1-t/(n-i))$이므로 반복횟수는 P_k^{-1}이 된다. 따라서 평균 워크팩터, T_{cc}는 $k^3 * P_k^{-1}$이 된다.

그러나 위에서의 평균 워크팩터는 그립터널리스트가 구한 M이 의미있는 메시지인지를 검증하는 워크팩터는 포함되지 않았다.

4.3 PRAC의 크립터널리스트 ATTACK

가. Known-plaintext attack

private-key 크립토시스템에서는 S, G, P와 마찬가지로 G*도 비밀 키로 유지된다. 따라서 크립터널리스트가 정확한 S, G, P를 추측하기란 거의 불가능하다. 따라서 크립터널리스트는 다음과 같은 2단계를 거치게 된다. [6]

단계 1. 많은 (M, C)pair를 가지고서 Z벡터를 추측을 해서 G*를 알아낸다.

단계 2. G*를 이용해서 C로 부터 M을 알아낸다.

예를들어 n=255, k=179, t=10일때 단계 1에서의 워크팩터 $T1 \approx 4.0 \times 10^{12}$이고 460,000 정도의 (M, C)pair가 필요하다.

따라서 PRAC은 크립터널리스트가 많은 (M, C)pairs를 확보하지 못하는 현실에서는 안전한 크립토시스템이라고 생각할 수 있다.

나. Chosen-Plaintext Attack

크립터널리스트는 임의로 선택한 많은 보통문과 암호문의 쌍을 갖고서 다음과 같은 두단계를 거치게 된다.

단계 1. 많은 (M, C)pairs를 가지고서 G*를 구한다.

단계 2. G*를 이용해서 C로부터 M을 구한다. (MPBC와 워크팩터가 같다.) 예를들어 n=255, k=179, t=10일때 단계 1에서의 work factor는 20개의 임의로 선택한 보통문-암호문(chosen plaintext-ciphertext)pairs로 약 10^6이 소요된다[6]. 따라서 Chosen-plaintext attack하에서는 PRAC은 MPBC와 거의 같은 크립토컴플렉시티를 갖는다.

4.4 원시이원(255, 179) BCH부호로의 확장

DES와 대등한 크립토컴플렉시티를 갖는 (255, 179) BCH부호를 사용하는 PRAC을 구현하기 위해서는 먼저 GF(2*)을 만들어야 한다.

m=8일때의 원시다항식은 $p(X)=1+X^2+X^3+x^4+x^8$이고 이 원시다항식으로 생성되는 Galois체 GF(2*)은 TABLE 1과 같다. TABLE 1에서의 각 element의 표현방법은

i $(a_0, a_1, \cdots, a_i)$에서 i는 α^i를 나타내고 이진 8-tuple은 $\alpha^i = a_0 + a_1\alpha + \cdots + a_8$을 나타낸다.

TABLE 1 $p(X) = 1+X^2+X^3+X^4+X^8$에 의해 생성되는 GF(2^8)의 ELEMENTS

	00000000	127	00110011
0	10000000	128	10100001
1	01000000	129	11101000
2	00100000	130	01110100
3	00010000	131	00111010
4	00001000	132	00011101
5	00000100	133	10110110
6	00000010	134	01011011
.	. . . .	.	. . . .
.	. . . .	.	. . . .
.	. . . .	.	. . . .
.	. . . .	.	. . . .
43	11101110	171	11001101
44	01110111	172	11011110
45	10000011	173	01101111
46	11111001	174	10001111
47	11000100	175	11111111
48	01100010	176	11000111
49	00110001	177	11011011
50	10100000	178	11010101
51	01010000	179	11010010
.	. . . .	.	. . . .
.	. . . .	.	. . . .
.	. . . .	.	. . . .
.	. . . .	.	. . . .
114	01111100	242	00001101
115	00111110	243	10111110
116	00011111	244	01011111
117	10110111	245	10010111
118	11100011	246	11110011
119	11001001	247	11000001
120	11011100	248	11011000
121	01101110	249	01101100
122	00110111	250	00110110
123	10100011	251	00011011
124	11101001	252	10110101
125	11001100	253	11100010
126	01100110	254	01110001

이때의 생성다항식 $g(X)$는 $1+X^2+X^3+X^5+X^8+X^{15}+X^{16}+X^{19}+X^{20}+X^{21}+X^{28}+X^{29}+X^{30}+X^{31}+X^{35}+X^{41}+X^{42}+X^{43}+X^{45}+X^{46}+X^{47}+X^{48}+X^{51}+X^{52}+X^{53}+X^{57}+X^{60}+X^{61}+X^{62}+X^{65}+X^{67}+X^{70}+X^{71}+X^{73}+X^{76}$가 된다. [3]

그리고 PRAC에서 키로 사용되는 S는 [179, 179]의 비 특이행열이되고 순열행열 P는 [255, 255] 행열이 된다. 이러한 S, G, P를 갖고서 PRAC을 구현하는 과정은 4장에서 기술된 알고리즘과 동일하다.

4.5 ACC에 대한 제한사항

가. BCH부호로 구현되는 ACC의 $G^*=SGP$에서 G는 G_3와 행 동치인 형태를 갖는데 크립터널리스트 입장에서 항상 G_3형태로서 사용된다. 물론 $m \geq 3$에서 각 차수마다 하나 이상의 원시다항식이 존재할 수 있으므로 G_3자체가 다를수 있다. 그러나 항시 다항식의 갯수는 ACC를 구현하는 BCH부호에서는 몇개 안되며[8] 크립터널리스트의 입장에서는 deterministic problem이다. 따라서 ACC에서 크립터널리스트는 G^*를 알고 있다고 할 수 있다.

나. MPBC의 크립터널리스트 Attack에서 G^*는 public키이고 G는 G_n이므로 $G^*=SG_nP$를 만족하도록 키를 factoring하면 암호문을 정확하게 해독할 수 있게 될 것이다. 반면 키

없이 C로부터 M을 구하는 5.2.의 (b)와 같은 방법의 워크팩터가 키를 factoring하는 것보다 다소 적다 하더라도 구한 보통문이 올바른 보통문인지를 판별하는데 더욱 많은 노력이 요구될 수 있다. 따라서 키의 factoring 방법이 MPBC의 더 좋은 Attack방법으로서 고려대상이 될 수 있겠다.

다. BCH부호로 MPBC를 구현시 $G^*=SG_nP$에서 만약 P가 G*의 열을 순회치환하는 순열행열이라면 P와는 다른 어떠한 순회 치환하는 순열행열 P_1이라하더라도 $G^*=S_1G_3P_1$을 만족하는 S_1과 함께 또다른 MPBC의 키가 된다.

TABLE 2 행 동치인 ACC의 키(key)

(4, 4) 비특이 행열 S	(7, 4) BCH 부호 생성 행열 G	(7, 7) 순열 행열 P	G′ = SGP
$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$	$\begin{bmatrix}1&1&0&1&0&0&0\\0&1&1&0&1&0&0\\1&1&1&0&0&1&0\\1&0&1&0&0&0&1\end{bmatrix}$	$\begin{bmatrix}1&0&0&0&0&0&0\\0&1&0&0&0&0&0\\0&0&0&0&0&0&1\\0&0&0&1&0&0&0\\0&0&0&0&1&0&0\\0&0&0&0&0&1&0\\0&0&1&0&0&0&0\end{bmatrix}$	$\begin{bmatrix}1&1&0&1&0&0&0\\0&1&0&0&1&0&1\\1&1&0&0&0&1&1\\1&0&1&0&0&0&1\end{bmatrix}$
$\begin{bmatrix}0&1&0&0\\1&0&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$	$\begin{bmatrix}1&1&0&1&0&0&0\\0&1&1&0&1&0&0\\1&1&1&0&0&1&0\\1&0&1&0&0&0&1\end{bmatrix}$	$\begin{bmatrix}0&0&0&0&0&0&1\\0&1&0&0&0&0&0\\1&0&0&0&0&0&0\\0&0&0&0&1&0&0\\0&0&0&1&0&0&0\\0&0&0&0&0&1&0\\0&0&1&0&0&0&0\end{bmatrix}$	$\begin{bmatrix}1&1&0&1&0&0&0\\0&1&0&0&1&0&1\\1&1&0&0&0&1&1\\1&0&1&0&0&0&1\end{bmatrix}$
$\begin{bmatrix}0&0&0&1\\0&1&0&0\\0&0&1&0\\1&0&0&0\end{bmatrix}$	$\begin{bmatrix}1&1&0&1&0&0&0\\0&1&1&0&1&0&0\\1&1&1&0&0&1&0\\1&0&1&0&0&0&1\end{bmatrix}$	$\begin{bmatrix}1&0&0&0&0&0&0\\0&0&0&0&0&0&1\\0&1&0&0&0&0&0\\0&0&1&0&0&0&0\\0&0&0&0&1&0&0\\0&0&0&0&0&1&0\\0&0&0&1&0&0&0\end{bmatrix}$	$\begin{bmatrix}1&1&0&1&0&0&0\\0&1&0&0&1&0&1\\1&1&0&0&0&1&1\\1&0&1&0&0&0&1\end{bmatrix}$

MPBC의 attack방법중 S, G_3, P를 factoring하기가 어렵다고 했는데 P가 만약 순회 치환행열이라면 암호시 사용된 키와 다른 n-2개의 키가 존재하여 그 만큼의 워크팩터가 감소하게 될 것이다.

PRAC에서도 P값이 순회 치환행열이라면 MG*의 값은 G_3의 row space내에 있게 되며 크립터널리스트는 쉽게 에러를 수정하여 known-plaintext attack하에서 C=MG*을 풀어서 G*를 구하고 S, G_3P값을 쉽게 factoring한다. 따라서 BCH부호로 ACC를 구현시 키로 사용되는 순열행열 P는 순회 치환행열이 되어서는 안된다.

라. G*=SG_3P에서 임의의 새로운 P_1이 있어서 G_3P_1이 G*와 행 동치(row equivalent)일 경우 G*=$S_1G_3P_1$을 만족하는 S_1이 존재하며 이때에 BCH부호로 구현되는 MPBC에서는 본래의 암호 키와 다른 키로 해독될 수 있고 TABLE 2에서는 그 예를 보여준다.

따라서 원시 이원 BCH부호로 구현되는 MPBC를 해독하려는 크립터널리스트는 public 키인 G*를 알고 있고 또한 생성행열 G가 G_3의 형태를 갖고 있고 라. 의 예에서와 같이 최초의 암호화 키와 다른 또 다른 키가 암호화 키가 되기 때문에 G*로부터 S, G, P의 도출시 McEliece가 제시한 4.2의 가. 의 내용보다는 적은 워크 팩터가 소요된다는 사실을 확인할 수 있다.

6. 결 론

이상에서 논술한 바와같이 본 논문에서는 지금까지의 통상적인 암호화 방법인 재배열방법, 치환방법 또는 이들을 복잡하게 반복 이용하는 혼합방법과는 성격이 다른 large distance algebraic부호를 사용한 ACC를 구현하기 위해서 원시 이원 BCH부호를 고찰하였고 PRAC을 구현하기 위해서 (15, 7) BCH부호를 예를들어서 상세히 알고리즘을 기술하였다. 또한 ACC에서 키로 사용되는 생성행열이 항상 조직 생성행열의 형태를 갖는다는 사실을 확인 하였고 MPBC에서 크립터널리스트 attack시 G*로부터 S, G, P의 도출방법이 키없이 직접 C로부터 M을 복구하는 방법보다 더 좋은 방법이 될 수 있다는 가능성을 제시하였으며 PRAC에서 키로 사용되는 순열행열 P가 순회 치환행열이 되지않아야 바람직하다는 내용을 제시하였다.

(255, 179) BCH부호로 구현되는 PRAC은 DES를 깨뜨리기위해 필요한 워크 팩터 $2^{56}\approx 10^{17}$ 보다 적은 워크팩터(n=255, k=179, t=10일때 $T1\approx 4.0\times 10^{12}$)를 필요로 하지만 약 48만개의 (M, C) pairs가 필요하다. 따라서 크립터널리스트가 이와같은 많은 (M, C) pairs를 획득하기 어려운 상황하에서는 안전한 크립토시스템이라 할 수 있는데[6] 이러한 (255, 179) BCH부호를 사용한 PRAC을 소프트웨어적으로 구현하기 위해서 본 논문에서는

(255, 179) BCH 부호의 생성행열과 GF(2^8)의 원소(elements)와 이를 구현하는 알고리즘을 제시하였다.

(255, 179) BCH부호를 사용한 PRAC의 처리속도를 빠르게 하기 위해서는 bit 연산을 효과적으로 할 수 있는 새로운 프로그램언어나 장비가 개발되어야 할 것이다.

현대의 정보체계는 컴퓨터와 밀접하게 연관되어있다. 이제는 누구나가 컴퓨터 범죄를 저지를 가능성이 높아졌으며 동시에 피해자가 될 가능성이 높아졌다. 이러한 차원에서 볼때 자료보안에 대한 보완책과 방지책이 시급히 요구된다. 본 논문에서는 언급이 안되었지만 키의 관리문제도 자료보안에서 대단히 민감한 문제가 되고 있다. 따라서 키 관리에 대한 연구도 지속적으로 이루어져야 하겠다.

참고문헌

〔1〕 Richard E. Blahut, Theory and Implementation of Error correcting Codes, Addison-Wesley, 1983, pp. 161-166.

〔2〕 E. R. Berlekamp, Algebraic Coding Theory, McGraw-Hill, New York, 1968.

〔3〕 Dorothy E. Denning, Cryptography and data security, Addison Wesley, 1982, pp. 1-4.

〔4〕 Shu Lin and Daniel J. Costello, Jr., Error Control Coding : Fundamentals and Applications Prentice-Hall, 1983, pp. 1-170.

〔5〕 R. J. McEliece, "A Public-Key Cryptosystem Based on Algebraic Coding Theory," DSN Progress Report, Jet Propulsion Laboratory, CA., Jan. & Feb. 1978, pp. 42-44.

〔6〕 Kil-Hyun Nam, "Cryptographic Models for Computer Communications," Ph. D. Dissertation, Dept. of Computer Science, University of Southwestern Louisiana, Fall, 1985.

〔7〕 T. Kasami, N. Tokura, Y. Iwadare, and Y. Inagaki, Coding Theory, Corona, Tokyo, 1974.

〔8〕 W. Wesley Peterson and E. J. Weldon, Jr., Error-Correcting Codes, Second edition, The MIT Press, 1972, pp. 476.

〔9〕 T. R. N. Rao, "Cryptosystems Using Algebraic Codes," Int'l. Conf. on Computer Systems Signal Processing, Bangalore, India, Dec. 1984.

〔10〕 이만영, 부호이론, 희중당, 1984, pp. 51.