

행렬을 이용한 오류정정부호의 설계[†]

허성훈*, 조성진**, 최연숙**, 황윤희**, 이성가**

*부경대학교 정보보호학(협), **부경대학교 수리과학부

Design of Error Correcting Code Using Matrix[†]

Seong-Hun Heo*, Sung-Jin Cho**, Un-Sook Choi**, Yoon-Hee Hwang**, Sung-Ga Lee**

*Dept. of Information Security, Graduate School, Pukyong National Univ.

**Division of Mathematical Sciences, Pukyong National Univ.

요약

컴퓨터 기술이 발전하고 초고속 유무선 통신망이 확대됨에 따라 효율적이고 신뢰할 수 있는 디지털 데이터통신 및 저장 시스템에 대한 요구가 증가하고 있다. 하지만 통신 채널에서 발생하는 오류를 효율적으로 제어하기 위해서 오류정정부호 장치가 디지털 데이터통신 및 저장 시스템을 설계할 때 매우 중요한 요소가 되었다. 본 논문에서는 특수한 행렬을 이용하여 검사비트를 생성하는 방법과 오류를 정정하는 방법을 제안한다.

1. 서론

오늘날 많은 양의 데이터가 다양한 컴퓨터 시스템과 서브시스템 사이에서 디지털 논리 회로와 상호 연결하여 전송된다. 시스템의 신뢰성은 회로 모듈 사이에서 데이터 전송의 무오류성(error-free)에 의존한다[1]. 하지만 전자적 잡음, 장치 결함, 시간 오류 등으로 인하여 시스템에는 항상 언제 발생할지 모르는 잠재적 오류가 존재한다[1][2]. 따라서 시스템의 신뢰성을 항상시키기 위해서 오류정정부호 장치가 필수 불가결한 요소가 되었다[3]. 일반적으로 오류정정부호는 (n, k, d) 부호로 나타내며, n 은 부호어의 길이, k 는 데이터의 길이, d 는 최소거리(minimum weight)를 나타낸다[2][3]. 기존의 오류정정부호는 k 값이 커짐에 따라 부호화 및 복호화 회로가 복잡해지는 단점이 있다[4]. 본 논문에서는 행렬을 이용하여 검사비트를 생성하고 오류가 포함된 수신어를 효과적으로 복호하는 방법을 제안한다. 본 논문의 구성은 다음과 같다. 2장에서는 단일 및 이중 비트 오류정정부호에 사용되

는 행렬을 구성하는 방법을 알아보고, 3장에서는 이 행렬을 이용하여 오류정정부호의 부호화 및 복호화 방법을 제안한다. 그리고, 4장에서 결론을 맺는다.

2. 오류정정부호의 설계

유한체 $GF(2)$ 상에서 다음 $k \times k$ 행렬 M_k 를 생각하자.

$$M_{ij} = \begin{cases} 0, & (1 < |i-j| < k-1) \\ a_{ij}, & (i=j) \\ l_{ij}, & (i=j+1 \text{이고}, j-i=k-1) \\ u_{ij}, & (i=j-1 \text{이고}, i-j=k-1) \end{cases} \quad (1)$$

여기서, $j-i=k-1$ 은 행렬 M_k 에서 $(1, k)$ 성분을 나타내며, $i-j=k-1$ 은 $(k, 1)$ 성분을 나타낸다. 따라서 행렬 M_k 는 다음과 같다.

$$M_k = \begin{bmatrix} a_1 & u_1 & 0 & 0 & \cdots & 0 & l_1 \\ l_2 & a_2 & u_2 & 0 & 0 & \cdots & 0 \\ 0 & l_3 & a_3 & u_3 & 0 & \cdots & 0 \\ \vdots & & & & & & \\ u_k & 0 & 0 & \cdots & 0 & l_k & a_k \end{bmatrix}$$

[†] 본 연구는 정보통신기초연구지원사업(03-기초-0047) 연구비에 의해 연구되었음.

[정리 1] 선형 (n, k) 부호 C 의 최소무게가 d 이기 위한 필요충분조건은 C 의 패리티 검사행렬 H 에서 서로 다른 $d - 1$ 개 이하의 열벡터가 독립이다[5]. \square

[정리 2] $k \times k$ 행렬 M 의 서로 다른 i ($0 < i < d$) 개의 열의 비트별 합이 적어도 $(d - i)$ 개의 1을 포함하면 행렬 M_k 은 최소거리가 d 인 부호를 생성한다. \square

[파름정리 1] 행렬 M 이 다음 두 조건을 만족한다면, $(n, k, 3)$ 부호를 생성한다:

- (a) M 의 모든 열벡터는 적어도 두 개의 1을 포함한다.
- (b) M 의 서로 다른 두 개의 열벡터는 적어도 하나의 위치에서 다르다. \square

[파름정리 2] 행렬 M 이 다음 두 조건을 만족한다면, CA는 $(n, k, 5)$ 부호를 생성한다:

- (a) M 의 모든 열벡터는 적어도 네 개의 1을 포함한다.
- (b) M 의 서로 다른 네 개 이하의 열벡터의 합은 영벡터가 아니다. \square

파름정리 1에 의해서 단일 비트 오류정정부호를 생성할 수 있으며, 파름정리 2에 의해서 이중 비트 오류정정부호를 생성할 수 있다.

3. 부호화 및 복호화 방법

(1) 부호화 방법

행렬 M_k 를 이용하여 오류정정부호를 생성하는 방법을 알아본다. 부호화의 첫 번째 과정은 파름정리 1(단일 오류정정부호)과 파름정리 2(이중 오류정정부호)를 만족하는 M 을 찾는 것이다. 수리 소프트웨어인 Mathematica를 이용하여 파름정리의 조건을 검사하는 프로그램을 작성하여 행렬 M 을 구성하였다. 알고리즘 I은 단일 오류정정부호를 위한 행렬 M 을 생성하는 알고리즘이다.

<알고리즘 I. 행렬 M 생성 알고리즘 >

Step 1. 행렬 M_k 를 구성한다.

Step 2. M_k 에서 같은 위치에서 1을 갖지 않는 행끼리 더하여 [파름정리 1]을 만족하는지 검사하여 행의 수를 최대한으로 줄인다.

Step 3. Step 2에서 구한 행렬을 M 으로 둔다.

알고리즘 II는 이중 오류정정부호를 생성하기 위한 행렬 M 을 구성하는 알고리즘이다.

<알고리즘 II. 행렬 M 생성 알고리즘 >

Step 1. 행렬 M_k 를 구성한다.

Step 2. M_k^3 을 구한다.

Step 3. M_k^3 의 $(1, k)$ 성분을 1로 바꾸고, 행렬 $[10101010\dots]$ $[01010101\dots]$ 을 덧붙인다.

Step 4. Step 3에서 구한 행렬을 M_k 라 하면 같은 위치에서 1을 갖지 않는 행끼리 더하여 [파름정리 2]를 만족하는지 검사하여 행의 수를 최대한으로 줄인다.

Step 5. Step 4에서 구한 행렬을 M 으로 둔다.

<예제 1> 알고리즘 II의 Step 1과 Step 2에서 구성된 M_{16} 과 M_{16}^3 은 각각 다음과 같다.

$$M_{16} = \begin{vmatrix} 1100000000000000 \\ 0110000000000000 \\ 0011000000000000 \\ 0001100000000000 \\ 0000110000000000 \\ 0000011000000000 \\ 0000001100000000 \\ 0000000110000000 \\ 0000000011000000 \\ 0000000001100000 \\ 0000000000110000 \\ 0000000000011000 \\ 0000000000001100 \\ 0000000000000110 \\ 0000000000000011 \\ 1000000000000011 \end{vmatrix}_{16 \times 16}$$

$$M_{16}^3 = \begin{vmatrix} 1111000000000000 \\ 0111100000000000 \\ 0011110000000000 \\ 0001111000000000 \\ 0000111100000000 \\ 0000011110000000 \\ 0000001111000000 \\ 0000000111100000 \\ 0000000011110000 \\ 0000000001111000 \\ 0000000000111100 \\ 0000000000011110 \\ 0000000000001111 \\ 10000000000000101 \\ 110000000000000000 \\ 011000000000000000 \end{vmatrix}_{16 \times 16}$$

Step 3에서 $(1, 16)$ 성분을 1로 바꾸고 행렬

$[1010101010101010]$ 을 연결하면,

$$M_k = \begin{vmatrix} 1111000000000000 \\ 0111100000000000 \\ 0011110000000000 \\ 0001111000000000 \\ 0000111100000000 \\ 0000011110000000 \\ 0000001111000000 \\ 0000000111100000 \\ 0000000011110000 \\ 0000000001111000 \\ 0000000000111100 \\ 0000000000011110 \\ 0000000000001111 \\ 1000000000000101 \\ 1100000000000000 \\ 0110000000000000 \\ 1010101010101010 \\ 0101010101010101 \end{vmatrix}_{18 \times 16}$$

이 된다. M_k 는 Step 4에 의해서 18행에서 11행으로 줄어든다. Step 5에서 M 은 다음과 같다.

$$M = \begin{vmatrix} 1111000000011111 \\ 0111100000011111 \\ 0011110011110000 \\ 1001111000000101 \\ 0000111100111100 \\ 0000011110000000 \\ 0110001111000000 \\ 0000000111100000 \\ 1100000001111000 \\ 1010101010101010 \\ 0101010101010111 \end{vmatrix}_{11 \times 16}$$

대한 검사비트의 크기를 나타낸 것이다.

데이터 (k)	검사비트($n-k$)
8	8
9	9
10	9
11	9
12	9
13	10
14	10
15	10
16	11
17	11
18	12
19	12
20	12
32	16

<표 3> 최소거리가 5인 검사비트

(2) 복호화 방법

수신된 부호어에서 오류가 있는 비트를 정정하기 위한 복호화 방법을 알아본다. 복호화하는 첫 번째 단계는 신드롬을 계산하는 것이다.

수신된 부호어가 $C' (= (D'|CB'))$ 라면 신드롬은 $S(C') = H C'^T = H [D'|CB']^T$ 이다. 여기서, H 는 알고리즘 I 과 II에 의해서 생성된 M 과 단위 행렬 I_k 를 연결하여 형성된 패리티 검사행렬이다. 즉, $H = [M | I_k]$ 이다.

따라서,

$$\begin{aligned} S(C') &= H [D'|CB']^T = [M | I_k] [D'|CB']^T \\ &= M D'^T \oplus I_k CB'^T = M D'^T \oplus CB'^T \end{aligned}$$

로 표현될 수 있다.

신드롬의 값이 0이면 오류가 없음을 나타내고 0이 아니면 수신된 부호어에 오류가 있음을 나타낸다.

수신된 부호어에서 D_e 과 CB_e 를 각각 데이터와 검사비트에 대응되는 오류 벡터라고 하자. 그러면, 수신된 데이터는 $D' = D \oplus D_e$ 이고, 수신된 검사비트는 $CB' = CB \oplus CB_e$ 이다.

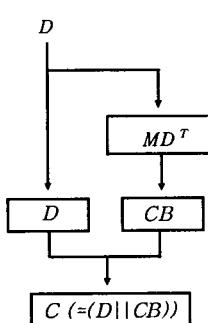
$$H [D | CB]^T \oplus H [D_e | CB_e]^T = S$$

여기서, $H [D | CB]^T = 0$ 이므로,

$$H [D_e | CB_e]^T = S$$

즉, $H [E]^T = [S]$ 에서 오류벡터 E 는

$[E]^T = H^{-1}[S]$ 이 된다. 하지만 H^{-1} 이 존재하



<그림 1> 부호화 방법

다음 표는 최소거리가 5인 부호에서 데이터에

려면 H 는 $n \times n$ 정방행렬이어야 한다. 하지만, H 가 정방행렬이 아니므로 패리티 검사 행렬 $H_{(n-k) \times n}$ 를 정방행렬 $M_{aug \ n \times n}$ 로 변환하기 위해서 k 개의 추가 행을 $H_{(n-k) \times n}$ 에 덧붙인다. 이때 $M_{aug \ n \times n}$ 가 역행렬이 존재하도록 구성한다. 즉, $\det[M_{aug}] = 1$ 이다.

$$\begin{aligned} M_{aug} &= \begin{bmatrix} [H]_{(n-k) \times n} \\ \vdots \\ [\text{추가 행}]_{k \times n} \end{bmatrix}_{n \times n} \\ M_{aug}[E]^T &= \begin{bmatrix} [H] \\ \vdots \\ [\text{추가 행}] \end{bmatrix}_{n \times n} [D_e | CB_e]^T \\ &= \begin{bmatrix} S \\ \vdots \\ S_{aug} \end{bmatrix}_{n \times 1} \end{aligned}$$

$$[E] = [M_{aug}]^{-1} \begin{bmatrix} S \\ S_{aug} \end{bmatrix}$$

<정의 1> 복호화에 사용되는 M_{aug} 는 알고리즘 II에 의해서 구성된 M 을 이용하여 다음과 같이 구성한다.

$$M_{aug} = \begin{bmatrix} M_{(n-k) \times k} & I_{n-k} \\ I_k & O_{k \times (n-k)} \end{bmatrix}_{n \times n}$$

□

[정의 3] 정의 1에서 구성된 M_{aug} 의 역행렬은 다음과 같다.

$$M_{aug}^{-1} = \begin{bmatrix} O_{k \times (n-k)} & I_k \\ I_{n-k} & M_{(n-k) \times k} \end{bmatrix}_{n \times n}$$

□

다음은 이중 오류정정부호의 복호화 알고리즘이다.

<알고리즘 III. 복호화 알고리즘>

Step 1. 신드롬 S 를 구한다.

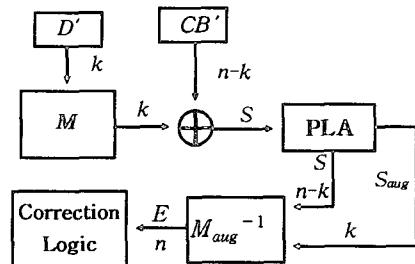
Step 2. M_{aug} 와 M_{aug}^{-1} 를 구한다.

Step 3. PLA를 이용하여 S_{aug} 를 대응시킨다.

Step 4. $E^T = M_{aug}^{-1} \begin{bmatrix} S \\ S_{aug} \end{bmatrix}$ 에 의해서 오류벡터 E 를 구한다.

Step 5. $C = C' \oplus E$ 에 의해서 부호어를 복호한다.

그림 2는 이중 오류정정부호의 복호화 방법을 나타낸다.



<그림 2> 복호화 방법

4. 결론

본 논문에서는 단일 및 이중 오류를 정정하기 위하여 행렬을 이용하여 오류정정부호를 구성하는 방법을 제안하였다. 단일 오류정정부호의 개념을 약간 변형하여 이중 오류정정부호까지 구성하는 방법을 제시하였다. 이 연구를 바탕으로 다중 오류정정부호의 설계에 대한 일반화 과정에 대한 연구가 필요할 것으로 사료된다.

참고문헌

- [1] P. P. Chaudhuri, et. al., "Additive cellular automata theory and applications", Vol. 1, IEEE Computer Society Press, California, USA, 1997.
- [2] T.R.N. Rao and E. Fujiwara, "Error-Control Coding for Computer System", Prentice Hall, Englewood Cliffs, N.J., 1989.
- [3] S. Lin and D.J. Costello, "Error Control Coding: Fundamentals and Applications", Prentice Hall, Englewood Cliffs, N.J., 1983.
- [4] D.R. Chowdhury, S. Basu, I.S. Gupta and P.P. Chaudhuri, "Design of CAECC - Cellular Automata Based Error Correcting Code", IEEE Trans. Computers, Vol. 43, June 1994, pp. 759-764.
- [5] F.J. MacWilliams and N.J.A. Sloane "The Theory of Error Correcting Codes", North-Holland, 1977.