# RS-부호에 유용한 3항 기약 다항식에서 새로운 TRACE 연산 알고리즘

서 창 호† 은 희 천††


## 요 약

이 논문에서는 데이타 통신에서 발생되는 오류를 정정하기 위해 많이 사용되고 있는 RS 부호의 3항 기약다항식에서 새로운 Trace연산 알고리즘에 대해 고찰한다. 이 방법은 기존의 방법에 비해 Trace를 간단한 연산으로 구할 수 있다. 이 새로운 알고리즘은 복잡한 연산을 피함으로써 연산시간을 줄일 수 있고, 복호화 과정을 간략히 할 수 있어서, 같은 정도의 데이터 신뢰도를 얻는데 요구되는 노력을 감소시킬 수 있다. 새로운 Trace 연산 알고리즘과 기존의 Trace 정의에 따른 방법은 SUN SPARC2 workstation상에서 C-언어로 구현한 결과를 비교,분석하였다.


# A New Trace Calculation Algorithm on Trinomial Irreducible Polynomial of RS code

Chang Ho Seo† and Hi Chun Eun††


## ABSTRACT

In this paper, we show that it is more efficient to use a new algorithm than to use a method of trace definition and property when we use trace calculation method on trinomial irreducible polynomial of reed-solomon code. This implementation has been done in SUN SPARC2 workstation using C-language.


## 1. 서 론

정보화 사회로 진입하고 있는 오늘날 유무선 통신상에서 보안유지및 관련문제는 효율성 문제와 함께 가장 시급하고 급속히 발전하고 있는 분야이다. 1960년 I.S.Reed 와 G.S.Solomon에 의해 만들어진 RS-부호는[1] 다중오류 정정부호인 BCH 부호[2]의 하나로서, 비2원 BCH부호의 한 범주에 속한다. 또, Galois체 $GF(p^m)$상의 심볼 단위로 부호화(encoding)및 복호화(decoding)[2, 3, 4]되므로, 통신의 채널상에서 발생되는 산발오류(random error)와 연집오류(burst error)를 모두 정정 할 수 있다. 따라서, RS-부호는 각종 디지탈 통신시스템 및 데이터 저장시스템의 신뢰성 향상 대책으로 광범위 하게 사용되고 있다. 그러나, RS-부호는 복호과정에서 오류의 위치뿐만 아니라, 2원 BCH 부호의 복호에서는 요구되지 않았던 오류치를 구해야 한다. 그래서, 오류 정정 능력이 커짐에 따라 복호기 회로의 구성이 복잡해질 수 있으나, 개선된 여러 가지 복호 알고리즘을 사용하면 복호화 장치의 복잡성을 극복할 수 있다. RS-부호를 복호화하는데 필요한, 유한체에서 표준기저(standard basis)[5, 6]의 Trace 연산이 필요한 경우, 지금까지는 Trace의 정의및 성질[7]을 이용하여 구했다. 그러나, 기존의 방식은 연산이 복잡하고 회로가 복잡할 뿐만 아니라, 복호화 시간을 지연시키는 요인중의 하나로 작용되어 왔다.

† 준 회 원: 고려대학교 수학과 박사과정
†† 비 회 원: 고려대학교 자연과학대학 수학과 교수
논문접수: 1994년 11월 4일, 심사완료: 1995년 1월 7일

이 논문에서는 연산을 복잡하게 하고, 회로의 구성을 어렵게 만드는 여러 가지 요소들을 충분히 고려하여, RS-부호에 유용한 새로운 Trace 연산 알고리즘을 제안 및 구현하고자 한다. 먼저, 가장 많이 사용되는, $m < 1000$의 조건을 만족하는 유한체에서 3항인 기약 다항식[8]의 새로운 Trace 연산 알고리즘을 소개하고, 이 연산 알고리즘을 기반으로 하여 계산 속도는 지연되나, 유한체에서 일반화된 m항인 기약 다항식[11]의 Trace 연산 알고리즘을 제안하고자 한다. 그리고 이 새로운 알고리즘이 얼마나 더 Trace 계산을 단순화 시킬 수 있고, 계산 속도를 감소 시키는지 비교하고자 한다. 다음장에서는 Trace 정의에 따른 방법의 이론적 근거가 되는 유한체와 Trace 정의 및 성질에 관하여 개략적으로 소개 한다. 그리고 3장에서는 본 논문에서 제안하는 새로운 Trace 연산 알고리즘에 대해 설명 하였으며, 4장에서는 제안한 각 알고리즘의 효율성을 Sun SPARC2 workstation상에서, C-언어로 구현한 각 알고리즘의 연산 시간을 측정하여 비교 분석하고, 마지막으로 5장에서는 결론 및 향후의 연구 방향을 고찰한다.

## 2. 유한체와 Trace 정의

### 2.1 유한체(Finite field)

$\alpha$를 $GF(2^m)$상의 원시원(primitive element)이라 할때, 0원(zero element)을 제외한 $2^m-1$개의 모든 유한체 원소(finite field element)들은 $\alpha$의 멱(power)으로 표시된다. 이때, 이 $\alpha$가 차수가 m인 원시다항식,

$$p(x) = p_0 + p_1x + \cdots + p_{m-1}x^{m-1} + x^m$$
$$p_i \in GF(2),\ 0 \le i \le m-1$$

의 근이면, $P(\alpha) = 0$이므로, $\alpha^m = p_0 + p_1\alpha + \cdots + p_{m-1}\alpha^{m-1}$가 되어 $GF(2^m)$상의 각 원소들은 차수가 $m-1$ 이하인 $\alpha$의 다항식들로 표현할 수 있다. 이와 같은 표현방법을 다항식 표현(polynomial representation)이라 하며, 이 다항식들의 계수들로 표현하는 방법을 벡터 표현(vector representation)이라 한다. 예로서, $GF(2^5)$상에서 원시다항식이 $p(x) = 1+x^2+x^5$이면, $GF(2^5)$상의 모든 원소들을 다항식 표현과 벡터 표현으로 기술할 수 있는데, 이는 〈표 1〉과 같이 표현된다.

〈표 1〉 $p(x) = 1+x+x^5$일 때의 갈로아체 $GF(2^5)$
〈Table 1〉 Galois field $GF2^5$) with $p(\alpha) = 1+\alpha+\alpha^5 = 0$

| 다항식 표현 | 벡터표현 | 다항식 표현 | 벡터표현 |
|---|---|---|---|
| 0 | 00000 | $\alpha^{15} = 1+\alpha+\alpha^2+\alpha^3+\alpha^4$ | 11111 |
| 1 | 10000 | $\alpha^{16} = 1+\alpha+\alpha^3+\alpha^4$ | 11011 |
| $\alpha^1$ | 01000 | $\alpha^{17} = 1+\alpha+\alpha^4$ | 11001 |
| $\alpha^2$ | 00100 | $\alpha^{18} = 1+\alpha$ | 11000 |
| $\alpha^3$ | 00010 | $\alpha^{19} = \alpha+\alpha^2$ | 01100 |
| $\alpha^4$ | 00001 | $\alpha^{20} = \alpha^2+\alpha^3$ | 00110 |
| $\alpha^5 = 1+\alpha^2$ | 10100 | $\alpha^{21} = \alpha^3+\alpha^4$ | 00011 |
| $\alpha^6 = \alpha+\alpha^3$ | 01010 | $\alpha^{22} = 1+\alpha^2+\alpha^4$ | 10101 |
| $\alpha^7 = \alpha^2+\alpha^4$ | 00101 | $\alpha^{23} = 1+\alpha+\alpha^2+\alpha^3$ | 11110 |
| $\alpha^8 = 1+\alpha^2+\alpha^3$ | 10110 | $\alpha^{24} = \alpha+\alpha^2+\alpha^3+\alpha^4$ | 01111 |
| $\alpha^9 = \alpha+\alpha^3+\alpha^4$ | 01011 | $\alpha^{25} = 1+\alpha^3+\alpha^4$ | 10011 |
| $\alpha^{10} = 1+\alpha^4$ | 10001 | $\alpha^{26} = 1+\alpha+\alpha^2+\alpha^4$ | 11101 |
| $\alpha^{11} = 1+\alpha+\alpha^2$ | 11100 | $\alpha^{27} = 1+\alpha+\alpha^3$ | 11010 |
| $\alpha^{12} = \alpha+\alpha^2+\alpha^3$ | 01110 | $\alpha^{28} = \alpha+\alpha^2+\alpha^4$ | 01101 |
| $\alpha^{13} = \alpha^2+\alpha^3+\alpha^4$ | 00111 | $\alpha^{29} = 1+\alpha^3$ | 10010 |
| $\alpha^{14} = 1+\alpha^2+\alpha^3+\alpha^4$ | 10111 | $\alpha^{30} = \alpha+\alpha^4$ | 01001 |

### 2.2 Trace 정의 및 성질

다음은 3.1절에 기술되는 유한체의 각 원소들에 대한 Trace 연산을 하기 위하여, 다음과 같은 유한체상에서 Trace 특성에 대해 고찰한다[5, 6].

[정의]

$GF(p^m)$내의 임의의 원소 $\beta$에 대한 Trace, $T_r(\beta)$는 다음과 같이 정의된다.

$$T_r(\beta) = \sum_{i=0}^{m-1} \beta^{p^i}$$

여기서, p는 소수(prime), m은 양의 정수(positive integer)이다.

이 Trace는 다음과 같은 성질을 갖는다.

[성질]

(1) Trace는 $GF(p)$에서 선형 사상(linear transformation)이다.
즉, $T\gamma(c\ \alpha + d\ \beta) = c,\ T\gamma(\alpha) + d$
$T\gamma(\beta)$, c, $d \in GF(p)$, $\alpha, \beta \in GF(p^m)$

(2) 임의의 원소 $\beta \in GF(p^m)$
$T\gamma(\beta^p) = [T\gamma(\beta)]^2 = T\gamma(\beta) \in GF(p)$

(3) $Tr(1) \equiv m(\text{mod } p)$

## 3. 유한체 $GF(2^m)$에서 3항인 기약 다항식의 Trace 연산 알고리즘

### 3.1 Trace 정의에 따른 방법

위의 2.2 절에서 보여준 정의 및 성질들을 이용하여, Trace 연산을 다음과 같이 계산할 수 있다. $\alpha$를 원시 다항식 $p(x) = x^5 + x^2 + 1$의 근이라 하고, $GF(2^5)$내의 원소 1, $\alpha$, $\alpha^2$, $\alpha^3$, $\alpha^4$, $\cdots$, $\alpha^{30}$에 대한 Trace 값을 구하여 보자.

$GF(2^5)$상에서 5차 기약 다항식 $p(x) = x^5 + x^2 + 1 \in GF(2)[x]$이면,

1) $T\gamma(1) \equiv 5(\text{mod } 2) = 1$
2) $T\gamma(\alpha) = \alpha + \alpha^2 + \alpha^4 + \alpha^8 + \alpha^{16} = 0$
3) $T\gamma(\alpha^2) = T\gamma(\alpha) = 0$
4) $T\gamma(\alpha^3) = \alpha^3 + \alpha^6 + \alpha^{12} + \alpha^{24} + \alpha^{48}$
$= \alpha^3 + \alpha^6 + \alpha^{12} + \alpha^{24} + \alpha^{17} = 1$
5) $T\gamma(\alpha^4) = [T\gamma(\alpha^2)]^2 = T\gamma(\alpha^2) = 0$
$\vdots$
31) $T\gamma(\alpha^{30}) = \alpha^{30} + \alpha^{60} + \alpha^{120} + \alpha^{240} + \alpha^{480}$
$= \alpha^{30} + \alpha^{29} + \alpha^{27} + \alpha^{23} + \alpha^{15} = 0$

이와 같은 방법으로 Trace 연산을 수행하는 경우, 원시 다항식들의 벡터표현을 전부 찾아야되는 번거로움이 수반되고, 컴퓨터로 이를 구현하면 복잡한 방법으로 인한 처리속도의 증가현상이 나타나는 단점이 있다. 따라서 본 논문에서는 Trace 정의에 따른 방법이 갖는 단점을 보완하기 위하여,다음절에 기술되는 바와같이 새로운 Trace 연산 알고리즘의 특성을 고찰했다.

### 3.2 새로운 Trace 연산 알고리즘

$GF(2^m)$상에서 m차인 기약 다항식 $p(x) = x^m + x^k + 1$ 라 하고, $j = m-k$라 정의하면,

Step 1 : $T\gamma(1) \equiv m(m \text{ mod } 2)$
Step 2 : $j$: *even* : $T\gamma(\alpha^i) = 0$ for $0 < i \le m-1$
Step 3 : $j$: *odd* : $T\gamma(\alpha^i) = \begin{cases} 1, \text{ if } j = n\ j\ (n \in Z) \\ 0, \textit{ otherwise} \end{cases}$

위에 기술한 각 step들의 알고리즘은 다음과 같다.

```
(새로운 Trace연산 알고리즘 : ITA)
  trace_tri(int m, int k, int Tr[ ])
  {
    int i, j;
    Tr[0] = m % 2;
    j = m- k;
    if((j % 2)==0) for (i=1;i<=m-1;i++) Tr[i] = 0;
      else
        for(i=1;i<=m-1;i++) if ((i % j)
        ==0) Tr[i] = 1;
          else
            Tr[i]=0;
```

본 연구과정을 통해 개발된 새로운 Trace 연산 알고리즘의 계산 과정에는 원시 다항식들의 벡터 표현이 요구되지 않고, 각 step 들을 계산 과정이 서로 독립적이므로, Trace 정의에 의한 방법을 사용하는 경우에 비해 연산 시간이 단축되는 특성을 보일 것으로 예측된다. 다음장에서는 계산 속도는 지연되나, 유한체에서 일반화된 m항인 기약 다항식에서 Trace 연산 알고리즘을 고찰한다.

### 3.3 유한체에서 m 항인 기약 다항식의 Trace 연산 알고리즘 일반화

$GF(2^m)$상에서 $p(x) = x^m + c_1x^{m-1} + c_2x^{m-2} + \cdots c_{m-1}x + c_m$을 m 차인 원시 다항식, $\{\alpha, \alpha^2, \alpha^3, \cdots \alpha^{m-1}\}$을 다항식들의 근이라고 할 때,

각 단계별 알고리즘은 다음과 같다.

Step 1 : $T\gamma(1) \equiv (\mathrm{mod}\ 2)$

Step 2 : $T\gamma(\alpha) + c_1 = 0$

Step 3 : $T\gamma(\alpha^2) + c_1\ T\gamma(\alpha) = 0$ ····· (*)

Step 4 : $T\gamma(\alpha) + c_1\ T\gamma(\alpha^{-1}) + \cdots + c_{j-1}\ T\gamma(\alpha) + [j \bmod 2]\ c_j = 0$

Step 5 : $T\gamma(\alpha^{m-1}) + c_1\ T\gamma(\alpha^{m-2}) + \cdots + c_{m-2}\ T\gamma(\alpha) + [(m-1) \bmod 2]\ c_{m-1} = 0$

(일반화된 Trace 연산 알고리즘)

```
trace_tri(int m, int c[ ],int tr[ ])
{
  int j , k;
  tr[0] = m % 2;
  for(j = 1 ; j<= ; j++ ) {
        tr[ j] = 0 ;
  for(k = 0 ; j< j-1 ; k++ ) {
        tr[ j] = tr[ j] - c[ k] * tr[ j-1-k] ;
          }
        tr[ j] = (tr[ j] + (j % 2 )*c[ j-1] % 2;
          } }
```

유한체에서 3항인 기약 다항식의 새로운 Trace 연산은 m 항인 기약 다항식의 새로운 Trace 연산 알고리즘에 적용할 경우 다음과 같다. 유한체에서 정의된 m 차 기약 다항식을 $p(x) = x^m + x^k + 1$라 하면,

$c_1, c_2, \cdots c_{m-k-1}, \cdots, c_{m-1}=0, c_{m-k}=1, c_m=1$ 이므로, 3.3절 (*)에서 제안된 알고리즘에 적용하면 바로,

$T\gamma(1) = m(m \bmod 2)$

$$T\gamma(\alpha) = \begin{cases} 1, \text{ if } i = n(m-k) \text{ and } (m-k)\ odd \\ 0, \ otherwise \end{cases}$$

임을 알수 있다.

이상과 같이 기술된 유한체에서 일반화된 m항인 기약 다항식의 새로운 Trace 연산 알고리즘은 Trace 정의에 의한 방법에 비해 그 연산 특성이 비교적 향상된 모습을 보이나, 각 step 들을 계산하려면 그 step 까지의 모든 계산이 선행되어야 하므로 비교적 많은 계산 시간이 요구된다.

## 4. Trace 연산 시뮬레이션 결과 및 논의

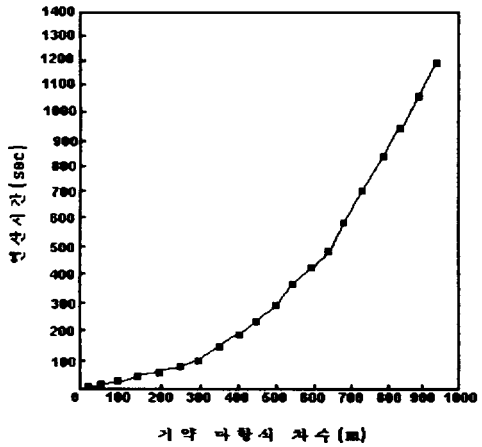
유한체 $GF(2^m)$상에서 m차인 원시 다항식의 최고차항의 차수 m=5라 하고, 다음 차수항의 차수 k=2 하면, $GF(2^5)$에서 원시 다항식 : $p(x) = x^5 + x^2 + 1$가 되고, m=7 이고 k=2라 하면 $GF(2^7)$에서 원시 다항식 : $p(x) = x^7 + x^2 + 1$, … 등 같이 된다. 이와같은 $GF(2^m)$상의 원시 다항식 $p(x)$에 대한 각 원소들의 Trace 연산 속도를 SUN SPARC2 workstation상에서 C-언어로 구현한 결과를 아래에 제시했다. 이 새로운 Trace 연산 알고리즘(ITA)은 Trace 정의에 따른 연산 방법(NFA)보다 현저하게 빠르고, 계산 과정을 단순화시킨다. $GF(2^m)$ 상에서 m 차인 원시 다항식 $p(x)=x^m + m^k + 1$라 할때, 〈표 2〉는 새로운 Trace 연산 알고리즘을 사용하는 경우, NFA 때보다 새로운 알고리즘의 계산속도가 월등히 빠름을 보여주고 있다.

(그림 1) 및 (그림 2)에 각각의 알고리즘(NFA, ITA)을 사용하여 Trace 연산을 수행하였을 때, 계산 시간(calculation time)을 기약 다항식의 차수 m으로 나타냈다. 〈표 2〉에서 보면, m 값이 10에서 950까지 증가함에 따라 NFA의 경

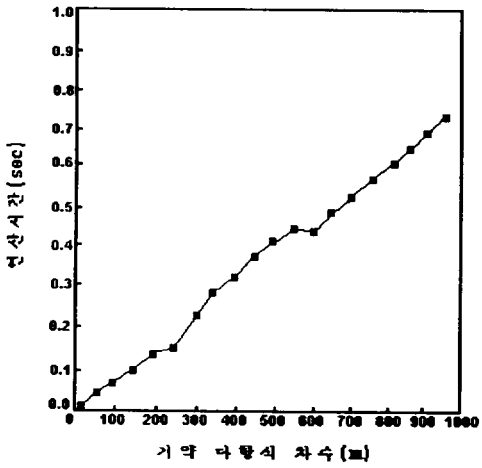
〈표 2〉 NFA 알고리즘과 ITA 알고리즘 계산 속도 비교
〈Table 1〉 compare NFA algorithm with ITA algorithm

| m | k | NFA | ITA |
|---|---|---|---|
| 10 | 2 | 0.148 | 0.012 |
| 50 | 2 | 3.351 | 0.043 |
| 100 | 2 | 13.186 | 0.076 |
| 150 | 2 | 29.670 | 0.109 |
| 200 | 3 | 52.747 | 0.153 |
| 250 | 3 | 81.168 | 0.164 |
| 300 | 4 | 118.130 | 0.236 |
| 350 | 4 | 160.980 | 0.274 |
| 400 | 4 | 209.890 | 0.307 |
| 450 | 2 | 265.380 | 0.346 |
| 500 | 2 | 327.470 | 0.390 |
| 550 | 2 | 395.600 | 0.423 |
| 600 | 3 | 467.030 | 0.424 |
| 650 | 3 | 554.945 | 0.500 |
| 700 | 2 | 637.362 | 0.538 |
| 750 | 2 | 736.263 | 0.576 |
| 800 | 2 | 840.000 | 0.615 |
| 850 | 2 | 945.054 | 0.650 |
| 900 | 3 | 1054.940 | 0.692 |
| 950 | 4 | 1181.310 | 0.730 |

우 계산시간이 0.148초에서 1181.310초로 급격히 증가하는 모습을 보인다. 그러나, ITA의 경우 계산시간이 0.012초에서 0.730초로, 계산속도의 증가폭이 아주 작음을 알 수 있다. 시뮬레이션을 통한 계산 결과을 제시한 바와 같이 m, k 값에 대해 새로운 알고리즘을 사용하여 계산한 Trace 연산 결과는 기존의 Trace 정의에 따른 방법을 사용하여 구하는 경우보다 훨씬 빠름을 알 수 있다. 즉, 연산 시간을 고려하면 기존의 알고리즘은 loop를 반복하면서 계산하므로 긴 시간이 요구되나, 반복되는 loop의 순환이 없는 새로운 알고리즘은 비교적 짧은 계산 시간을 요구하는 우수한




(그림 1) NFA 알고리즘 계산 시간
(Fig. 1) Execution times for NFA algorithm




(그림 2) ITA 알고리즘 계산 시간
(Fig. 2) Execution times for ITA algorithm

특성을 보여주고 있다. 또한, m 값이 증가함에 따라 기존의 Trace 정의에 따른 방법과 새로운 Trace 연산 알고리즘의 처리속도 차는 현저히 큰 폭을 나타내고 있음을 알 수 있다. 따라서 본 논문에 제시한 새로운 알고리즘을 사용하면, 연산 속도 증가와 더불어, 단일 칩(chip)으로 간단히 연산회로를 구성할 수 있어서, 더욱 빠른 처리속도를 기대할 수 있다.

## 5. 결론 및 향후 연구 방향

통신에 사용되는 RS-부호는 연집오류(burst error)의 정정에 많이 사용되는데, 복호화 과정이 복잡하므로 이를 개선하기 위한 다수의 연구가 진행되고 있다. 즉, 유한체에서 표준 기저의 Trace 연산이 필요한 이 Trace 연산 과정이 비효율적이고 긴 연산 시간이 요구되어, 그 영향은 복호화 과정 뿐만 아니라, 통신 시스템 전반에 바람직하지 못한 결과를 초래한다. RS-부호의 복호화에 유용한 3항인 기약 다항식에서 m 값이 증가할수록, 기존의 Trace 정의에 따른 방법과 새로운 Trace 연산 알고리즘의 처리 속도 차는 더욱 큰 폭을 보였다. 따라서, 새로운 알고리즘의 사용으로 연산 속도의 증가와 더불어, 단일 칩(chip)으로 간단히 연산 회로를 구성할 수 있어서, 더욱 빠른 처리속도를 기대할 수 있다. 그리고, m항인 원시 다항식의 일반적인 Trace 연산 알고리즘에 대한 지속적인 연구를 통해 그 개선 방향을 모색하고자 한다. 연구하고자하는 Trace 연산 알고리즘은 연산의 복잡성을 극복하는 방향을 제시할 것으로 예측되므로, 통신 시스템 전체를 구성하는 각 요소들 가운데 불필요한 요소를 상당히 제거할 수 있다고 생각 된다. 따라서, 시스템을 구성하는 전체 비용 가운데, 불필요한 연산에 들어가는 오버헤드(overhead)를 대폭 줄일 수 있을 것으로 기대된다.

## 참 고 문 헌

[1] E. R. Berlekamp, "Bit-serial Reed-Solomon

Encoders" IEEE Trans. Inf. Theory. Vol. IT-28, No. 6, pp. 869-874, 1960.

[2] G. D. Forney, "On Decoding BCH Codes" IEEE Trans. Inf. Theory, Vol. IT-11, pp. 577-580, 1965.

[3] D. M. Mandelbaum, "Decoding beyond the Designed Distance for Certain Algebraic Codes" Inf. and contr., pp. 209-228, 1977.

[4] D. C. Gorenstein and W. W. Piterson, "Polynomial Codes over Certain Finite Field" J. Soc. Ind. Appl. Math., Vol. 8, pp. 300-304, 1960.

[5] R. Lidl and H. Niederreiter, "Finite Fields" Addison-Wesley Publishing Company, 1983.

[6] T. Bartee and D. Schneider, "Computation with Finite Fields" Inf. and Contr., Vol. 6, pp. 79-98, 1963.

[7] I. S. Reed and T. K. Truong, "The Use of Finite Fields to Compute Convolutions" IEEE Trans. Inf. Theory, Vol. IT-21, pp. 206-213, 1975.

[8] N. Zierler and J. Brillhart, "On Primitive Trinomials(Mod 2)" Inf. and Contr., Vol. 13, pp. 541-554, 1968.

[9] F. J. mac Williams and N. J. A. Sloane, "The Theory of Error-Correcting Codes North-Holland", 1977.

[10] R. A. Rueppel and O. Staffelbach, "Linear Recurring Sequences with maximum Complexity", IEEE Trans. Inf. Theory, Vol. IT-33, pp. 126-131, 1987.

[11] T. W. Hungerford, "Algebra", Springer-Verlag, 1984.

**서 창 호**

1990년 고려대학교 수학과 졸업(학사)
1992년 고려대학교 대학원 수학과(이학석사)
1993년~현재 고려대학교 대학원 수학과 박사과정 수료
관심분야: 응용 대수학, 암호학, 정보 통신(coding).

**은 희 천**

1969년 고려대학교 수학과 졸업(학사)
1974년 고려대학교 대학원 수학과(이학석사)
1982년 고려대학교 대학원 수학과(이학박사)
1976년~80년 숭전대학교 수학과 조교수
1981년~현재 고려대학교 수학과 교수
관심분야: 해석학, 확률론.