Assignment: McEliece Cryptosystems, Due Wednesday, 11:59pm

Chris Betsill

December 7, 2016

1. Random Linear Codes

An $[n, k]$ *random linear code* is a binary linear code with generating matrix of the form $\begin{bmatrix} I_k & P \end{bmatrix}$, where the rows of P are $2i$-dimensional binary vectors of weight i, chosen at random, where $2i = n - k$.

a. Explain why the distance of such a code cannot exceed $(n - k)/2 + 1$.

Since each row of P has a weight of i, and $i = \frac{n-k}{2}$, and I_k always has a weight of 1, the distance of the linear code cannot exceed $i + 1$, which is equal to $\frac{n-k}{2} + 1$

b. Find a generating matrix for a $[21, 5]$ random linear code with distance 9. Show how you produced your example.

```
vec <- vector()
i <- 2
vec <- c(vec, randBinVector(21, sample(1, 1:27)))
temp <- vector()
C <- matrix(vec, nrow = 1, byrow = TRUE)
while(i < 6){
  if(i < 4){
    test <-  randBinVector(21, 13)
    temp <- c(vec, test)
    tempC <- matrix(temp, nrow = i, byrow = TRUE)
    if(codeDistance(tempC) >= 9){
      vec <- c(vec, test)
      i <- i + 1
    }
  }
  else{
    test <-  randBinVector(21, 12)
    temp <- c(vec, test)
    tempC <- matrix(temp, nrow = i, byrow = TRUE)
    if(codeDistance(tempC)  == 9){
      vec <- c(vec, test)
      i <- i + 1
    }
  }
}
C <- matrix(vec, nrow = 5, byrow = TRUE)


codeDistance(C)
```

```
[1] 9
```

```
generatorMatrix(C)
```

```
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
 [1,]    1    0    0    0    0    0    0    0    0     0     0     0     0
 [2,]    0    1    0    0    0    0    0    0    0     0     0     0     0
 [3,]    0    0    1    0    0    0    0    0    0     0     0     0     0
 [4,]    0    0    0    1    0    0    0    0    0     0     0     0     0
 [5,]    0    0    0    0    1    0    0    0    0     0     0     0     0
 [6,]    0    0    0    0    0    1    0    0    0     0     0     0     0
 [7,]    0    0    0    0    0    0    1    0    0     0     0     0     0
 [8,]    0    0    0    0    0    0    0    1    0     0     0     0     0
 [9,]    0    0    0    0    0    0    0    0    1     0     0     0     0
[10,]    0    0    0    0    0    0    0    0    0     1     0     0     0
[11,]    0    0    0    0    0    0    0    0    0     0     1     0     0
[12,]    0    0    0    0    0    0    0    0    0     0     0     1     0
[13,]    0    0    0    0    0    0    0    0    0     0     0     0     1
[14,]    0    0    0    0    0    0    0    0    0     0     0     0     0
[15,]    0    0    0    0    0    0    0    0    0     0     0     0     0
[16,]    0    0    0    0    0    0    0    0    0     0     0     0     0
      [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21]
 [1,]     0     0     0     1     0     1     1     1
 [2,]     0     0     0     0     1     1     0     0
 [3,]     0     0     0     0     0     1     1     0
 [4,]     0     0     0     0     1     1     1     0
 [5,]     0     0     0     0     1     0     1     0
 [6,]     0     0     0     0     0     1     1     1
 [7,]     0     0     0     0     1     1     1     1
 [8,]     0     0     0     0     1     1     0     1
 [9,]     0     0     0     0     1     0     1     1
[10,]     0     0     0     0     0     0     1     0
[11,]     0     0     0     0     0     0     0     1
[12,]     0     0     0     0     1     1     0     1
[13,]     0     0     0     0     1     1     1     1
[14,]     1     0     0     0     1     1     1     0
[15,]     0     1     0     0     1     0     0     1
[16,]     0     0     1     0     1     0     0     1
```

c. Does syndrome decoding offer any advantage over brute force nearest neighbor decoding for this code? Explain.

No,syndrome decoding has the same number of operations as brute force nearest neighbor. ## 2. McEliece cryptosystem

Illustrate the seven steps of the McEliece public key cryptosystem using the code generated by the matrix `bch15` from Problem 1 of the last assignment. Make and state all of the required choices: the message Alice wants to send, the choices Bob makes to obtain the public key, and the random vector Alice uses. Use the functions `randBinVector`, `randInvMatrix`, and `randPermMatrix` appropriately. Show the results of all the intermediate calculations, and confirm that decoding works properly.

1. Bob chooses the code generated by the matrix below, which can correct 3 errors.

```
G <- matrix(c(1,0,0,0,0,1,1,1,0,1,1,0,0,1,0,
              0,1,0,0,0,0,1,1,1,0,1,1,0,0,1,
              0,0,1,0,0,1,1,0,1,0,1,1,1,1,0,
              0,0,0,1,0,0,1,1,0,1,0,1,1,1,1,
              0,0,0,0,1,1,1,0,1,1,0,0,1,0,1),
            byrow=TRUE, nrow=5)
n <- ncol(G)
k <- nrow(G)
t <- 3
```

2. He then chooses a random binary invertible $k \times k$ matrix S and a random $n \times n$ permutation matrix P.

```
S <- randInvMatrix(k)

S[[1]]
```

```
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    0    1    1    1
[2,]    1    1    1    0    0
[3,]    1    1    0    0    0
[4,]    1    1    1    0    1
[5,]    0    0    0    1    0
```

```
P <- randPermMatrix(n)

P
```

```
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
 [1,]    0    0    0    0    0    1    0    0    0     0     0     0     0
 [2,]    0    1    0    0    0    0    0    0    0     0     0     0     0
 [3,]    0    0    0    1    0    0    0    0    0     0     0     0     0
 [4,]    0    0    1    0    0    0    0    0    0     0     0     0     0
 [5,]    0    0    0    0    0    0    0    0    1     0     0     0     0
 [6,]    0    0    0    0    0    0    0    0    0     0     0     1     0
 [7,]    0    0    0    0    0    0    1    0    0     0     0     0     0
 [8,]    0    0    0    0    0    0    0    0    0     0     0     0     0
 [9,]    0    0    0    0    0    0    0    0    0     0     1     0     0
[10,]    0    0    0    0    0    0    0    0    0     0     0     0     0
[11,]    0    0    0    0    0    0    0    0    0     1     0     0     0
[12,]    0    0    0    0    0    0    0    0    0     0     0     0     1
[13,]    0    0    0    0    0    0    0    1    0     0     0     0     0
[14,]    0    0    0    0    1    0    0    0    0     0     0     0     0
[15,]    1    0    0    0    0    0    0    0    0     0     0     0     0
      [,14] [,15]
 [1,]     0     0
 [2,]     0     0
 [3,]     0     0
 [4,]     0     0
 [5,]     0     0
 [6,]     0     0
 [7,]     0     0
 [8,]     0     1
```

```
 [9,]     0     0
[10,]     1     0
[11,]     0     0
[12,]     0     0
[13,]     0     0
[14,]     0     0
[15,]     0     0
```

3. Bob publishes $G_1 = SGP$ and keeps S, G, and P secret.

```
G1 <- ((S[[1]]%*%G)%*%P) %%2 # public key
G1
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
[1,]    0    0    1    1    1    1    0    1    1     0     0     1     0
[2,]    1    1    0    1    0    1    1    1    0     1     0     0     0
[3,]    1    1    0    0    1    1    0    0    0     0     1     1     1
[4,]    0    1    0    1    0    1    0    0    1     1     1     1     0
[5,]    1    0    1    0    1    0    1    1    0     0     0     0     1
     [,14] [,15]
[1,]     1     0
[2,]     1     0
[3,]     1     0
[4,]     0     0
[5,]     1     1
```

4. Alice encodes her message as a k-dimdimensional row vector (in this case randomly generated), and generates a random n-dimdimensional error vector e with weight t, and sends $y = xG_1 + e$

```
e <- randBinVector(n, t)

e
```

```
 [1] 0 1 0 0 0 0 0 0 1 0 0 0 0 0 1
```

```
Message <- randBinVector(k, 4)

Message
```

```
[1] 1 0 1 1 1
```

```
y <- ((Message%*%G1) + e)%%2

y
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
[1,]    0    1    0    0    1    1    1    0    1     1     0     1     0
     [,14] [,15]
[1,]     1     0
```

5. To decrypt, Bob calculates $y_1 = yP^{-1}$

```
y1 <- (y %*% solve(P) ) %% 2

y1
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
[1,]    1    1    0    0    1    1    1    0    0     1     1     0     0
     [,14] [,15]
[1,]     1     0
```

6. He then uses syndrome decoding to decode y_1 to obtain the code word x_1. The first k bits are x_0

```
cs <- syndrome(y1, parityCheckMatrix(G))

clst <- cosetLeaderSyndromeTable(G)
synds <- names(clst)
x1 <- clst[[match(cs, synds)]]

x0 <- (y1+x1) %% 2

x0 <- x0[1:k]

x0
```

```
[1] 1 0 0 0 0
```

7. Finally, he gets the decrypted message by calculating $x_0 S^{-1}$

```
code <- (x0 %*% S[[2]]) %% 2

code
```

```
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    0    1    1    1
```

```
print(all(code == Message))
```

```
[1] TRUE
```

3. McEliece with Goppa Codes

Suppose Bob implements a McEliece cryptosystem using a Goppa code with parameters $m = 7$ and $t = 15$.

a. How long is this code? How many errors can it correct? How many code words are there?

This code is $2^7 = 128$ bits long, and it can correct 15 codewords, since the minimum distance between codes is $2t + 1$. There are 23 codes.

b. When Alice sends a message, she chooses a random error vector e. How many different possible error vectors can she choose from?

```
n <- 2^7
possibleE <- choose(n, n-15)

possibleE
```

```
[1] 1.321671e+19
```

c. Suppose that Eve attempts a brute force attack to find the plaintext x by calculating all possible ciphertext messages that Alice could send. How many different ciphertexts of the form $y = xG_1 + e$ are there, as x ranges over all possible plaintexts and as e ranges over all possible error vectors? (Note that even though this number seems large, the parameters given in this example are too small to be used in practice.)

```
possibleM <- 2^n

possibleM* possibleE
```

```
[1] 4.497414e+57
```

4. Identify this cyclic code

a. Show that $x^2 + x + 1$ is a factor of $x^3 + 1$.

$\frac{x^2+x+1}{x^3+1}$

$x(x^2 + x + 1) = x^3 + x^2 + x$

$(x^3 + 1) - (x^3 + x^2 + x) = x^2 + x + 1$

$\frac{x^2+x+1}{x^2+x+1} = 1$

Thus, $\frac{x^2+x+1}{x^3+1} = x + 1$

b. Find all the code words of the cyclic code corresponding to the ideal $\langle x^2 + x + 1\rangle$ of $\mathbb{Z}_2[x]/(x^3 + 1)$. Write these code words as binary strings. Which familiar code is this?

$x^2 + x + 1 = 111$

Any multiplication of x to this always results in $x^2 + x + 1$

Thus the code words are

111

000

This is a 3-repetiton code.

5. Parity check polynomial

Consider the cylic code corresponding to the ideal $\langle x + 1\rangle$ in $\mathbb{Z}_2[x]/(x^4 + 1)$.

a. Find a parity check polynomial for this code.

$\frac{x+1}{x^4+1}$

$x^3 \times x + 1 = x^4 + x^3$

$(x^4 + 1) - (x^4 + x^3) = x^3 + 1$

$\frac{x+1}{x^3+1}$

$x^2 \times x + 1 = x^3 + x^2$

$(x^3 + 1) - (x^3 + x^2) = x^2 + 1$

$\frac{x+1}{x^2+1}$

$x \times (x + 1) = x^2 + x$

$(x^2 + 1) - (x^2 + x) = x + 1$

$\frac{x+1}{x^2+1} = 1$

Thus $\frac{x+1}{x^4+1} = x^3 + x^2 + x + 1$

b. Use your parity check polynomial to determine whether each of the following is a code word:

- $1 + x + x^3$

$\times x^3 + x^2 + x + 1$

$= x^6 + x^5 + 2x^4 + 3x^3 + 2x^2 + 2x + 1$

$= x^2 + x + 0 + x^3 + 0 + 0 + 1$

$= x^3 + x^2 + x + 1$

$1 + x + x^3$ is not a codeword

- $x + x^2$

$\times x^3 + x^2 + x + 1$

$= x^5 + 2x^4 + 2x^3 + 2x^2 + x$

$= x + 0 + 0 + 0 + x$

$= 0$

$x + x^2$ is a codeword