

# An Overview of the McEliece Cryptosystem with Mathematica Implementation

George Leibowitz

## ABSTRACT

Data security is a rising topic that is becoming increasingly important with the integration of various “smart” devices into our lives. The ideas of data security in the most basic and fundamental underlying form lead me to exploring the subject of data encryption. After learning the basics of data security through research, I was led to the McEliece Cryptosystem. Although various types of cryptosystems exist, the McEliece Cryptosystem was one that was not as utilized as others in the field, which was quite interesting for me, as I saw great potential in the system. Although the patterns that come from this system are finite due to the design of the system, the number of patterns is overwhelming. I decided to start this research project to replicate the McEliece Cryptosystem using Mathematica so that I could alter the system's inputs and parameters and really explore how the security was created. Through the coding process, which involved plenty of troubleshooting and revisions, I could deeply get to understand the logic behind the system. The McEliece Cryptosystem could also be important in the future for quantum cryptography, a field that will expand when quantum computing has become mainstream.

## 1. INTRODUCTION

This paper discusses the fundamental concepts of the McEliece Cryptosystem; an asymmetric encryption algorithm developed in 1978 by Robert McEliece and the first such scheme to use randomization in the encryption process. The algorithm has never gained much acceptance in the cryptographic community. As the system is immune to attacks using Shor's algorithm and measuring cost states using Fourier sampling, the McEliece Cryptosystem is a valid candidate. The algorithm is based on the hardness of decoding a general linear code, which is known to be NP hard. The original algorithm uses binary Goppa codes (subfield codes of geometric Goppa codes of a genus-0 curve over finite fields of characteristic 2); these codes are easy to decode, thanks to the efficient Patterson Algorithm. [3] The public key is derived from the private key by disguising the selected code as a general linear code. For this, the code's generator matrix  $G$  is perturbed by two randomly selected invertible matrices  $S$  and  $P$ . Variants of this cryptosystem exist, using different types of codes. Most of them were proven less secure, as structural decoding broke them. On the other hand, Yongge Wang proposed a secure McEliece scheme based on any efficient linear code; the hardness of Wang's variants depend on the NP-hardness of decoding random linear code. [1] The McEliece Cryptosystem with Goppa codes has resisted cryptanalysis until now. The most effective attacks known use information-set decoding algorithms. A 2008 paper by Daniel J Bernstein, Tanja Lange, and Christiane Peters describes both an attack and a fix. [2] Another paper by

christiane Peters shows that for quantum computing, key sizes must be increased by a factor of four due to improvements in information set decoding. [3]

The McEliece Cryptosystem has some advantages over, for example, RSA. The encryption and decryption are faster (for comparative benchmarks see the eBATS benchmarking project at [bench.cr.yp.to](http://bench.cr.yp.to)). For a long time, it was thought that McEliece could not be used to produce signatures. However, a signature scheme can be constructed based on the Niederreiter scheme, the dual variant of the McEliece scheme. Another advantage of the cryptosystem was mentioned before; theoretically it seems to have no vulnerabilities to quantum computing. One of the main disadvantages of the McEliece Cryptosystem is that the private and public keys are large matrices. For a standard selection of parameters, the public key is 512 kilobits long, leading to the rare use in practice. One exceptional case that uses the McEliece Cryptosystem for encryption is the Freenet-like application Entropy. Cryptography is a very important subject, as the World Wide Web, consisting of millions of interconnected computers and databases, allows nearly instantaneous communication and transfer of information all around the world. Cryptography makes secure web sites and all types of secure electronic transmissions possible. The contributions of cryptography allow people to conduct business and communications without worries of deception and theft. Every day, millions of people interact over the World Wide Web, with connections such as social media or e-commerce. The exponential increase of information on the World Wide Web has led to an increased reliance on cryptography.

Recently, a very large issue of privacy was raised when Apple went against the FBI's decision to force Apple to unlock the phone of a suspect of the San Bernardino murder case. [4] Apple denied, responding with the statement, "All that information needs to be protected from hackers and criminals who want to access it, steal it, and use it without our knowledge or permission." Undeniably, a lot of personal information has been uploaded to individual accounts all around the world, and the importance of keeping this type of information secure has become an essential challenge for the world today.

## 2. PRELIMINARIES

The original purpose of coding theory is to ensure that the transmission of information is correct. When the messages are sent in binary, some of the bits may become "flipped" to the opposite value (a zero to one or one to zero). Additional bits can be attached to each message so that the receiver can detect and correct the errors that occur. The main problem in this situation is to find efficient ways to attach additional information to messages so that the addressee can correct as many errors as possible. Definition 1.. Let  $V = \{b_1, b_2, \dots, b_n \mid b_i \in \{0,1\}\}$ , where addition in  $V$  is a component-wise addition modulo 2 ( $\oplus$  operation).  $V$  is called binary linear code if the next conditions hold

1.  $V$  includes trivial sequence.
2. For any pairs  $b_i, b_j$  their bitwise sum  $b_i \oplus b_j$  also belongs to  $V$ . Let's recall that  $0 \oplus 0 = 0$ ;  $0 \oplus 1 = 1$ ;  $0 \oplus 1 = 1$ ;  $1 \oplus 1 = 0$  [3]. Example 1 Let's  $b_1 = 10011$ ,  $b_2 = 01010$ , then

Remark, that this action can be performed with help of Mathematica (see [4]): `BitXor[{1,0,0,1,1},{0,1,0,1,0}]{1,1,0,0,1}`

### Exercise 1

Find  $b_4 \oplus b_3$  if  $b_3 = 00101$ ;  $b_4 = 10110$ .

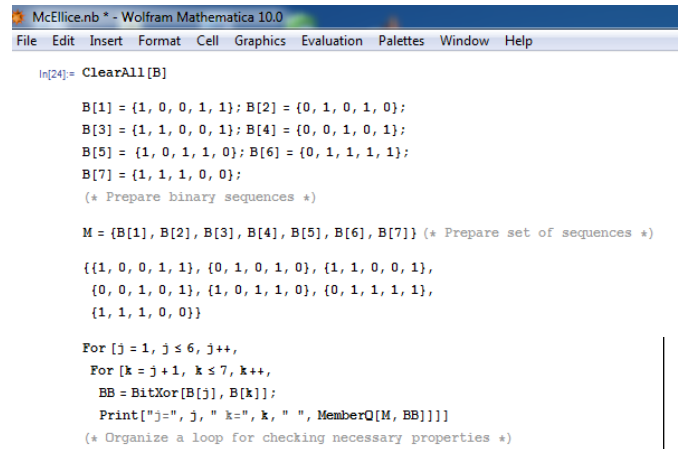
*Solution 1:*  $b_4 \oplus b_3 = 10010$ .

### Example 2

Generate  $V = \{b_0, b_1, \dots, b_7\}$  where  $b_0 = 00000$ ;  $b_5 = 11001$ ,  $b_6 = 01111$ ,  $b_7 = 11100$  and  $b_1, \dots, b_4$  are as in example 1 and exercise 1. Let's check that  $V$  is binary linear code with help of Mathematica.

To do this we organize a cycle for find  $\sum b_k \wedge b_j$  for all pairs  $\{k, j\}$ . (see fig.1).

The result of this code performing shows us that  $V$  is actually binary linear code (see fig.2).



```

McEllice.nb - Wolfram Mathematica 10.0
File Edit Insert Format Cell Graphics Evaluation Palettes Window Help

In[24]:= ClearAll[B]

B[1] = {1, 0, 0, 1, 1}; B[2] = {0, 1, 0, 1, 0};
B[3] = {1, 1, 0, 0, 1}; B[4] = {0, 0, 1, 0, 1};
B[5] = {1, 0, 1, 1, 0}; B[6] = {0, 1, 1, 1, 1};
B[7] = {1, 1, 1, 0, 0};
(* Prepare binary sequences *)

M = {B[1], B[2], B[3], B[4], B[5], B[6], B[7]} (* Prepare set of sequences *)

{{1, 0, 0, 1, 1}, {0, 1, 0, 1, 0}, {1, 1, 0, 0, 1},
{0, 0, 1, 0, 1}, {1, 0, 1, 1, 0}, {0, 1, 1, 1, 1},
{1, 1, 1, 0, 0}}

For[j = 1, j <= 6, j++,
  For[k = j + 1, k <= 7, k++,
    BB = BitXor[B[j], B[k]];
    Print["j=", j, " k=", k, " ", MemberQ[M, BB]]]]
(* Organize a loop for checking necessary properties *)

```

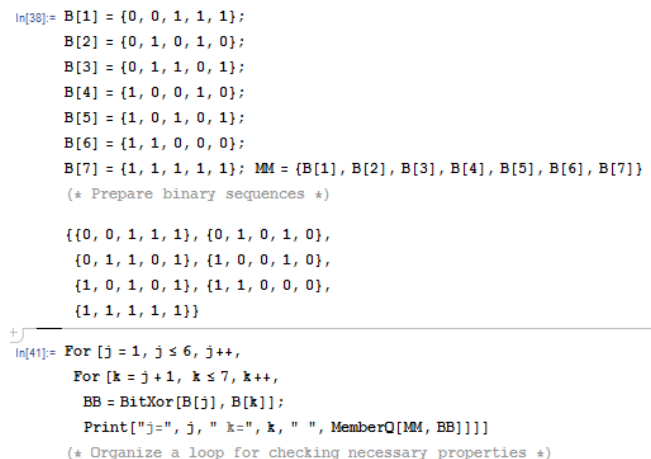
Figure 1. 1st fragment of Mathematica code

|              |              |              |
|--------------|--------------|--------------|
| j=1 k=2 True | j=2 k=4 True | j=3 k=7 True |
| j=1 k=3 True | j=2 k=5 True | j=4 k=5 True |
| j=1 k=4 True | j=2 k=6 True | j=4 k=6 True |
| j=1 k=5 True | j=2 k=7 True | j=4 k=7 True |
| j=1 k=6 True | j=3 k=4 True | j=5 k=6 True |
| j=1 k=7 True | j=3 k=5 True | j=5 k=7 True |
| j=2 k=3 True | j=3 k=6 True | j=6 k=7 True |

Figure 2. The result of executing the 1st fragment of Mathematica code Exercise 2

Check, that  $V = \{b_0, b_1, \dots, b_7\}$  where  $b_0 = 00000$ ,  $b_1 = 00111$ ,  $b_2 = 01010$ ,  $b_3 = 01101$ ,  $b_4 = 10010$ ,  $b_5 = 10101$ ,  $b_6 = 11000$ ,  $b_7 = 11111$  is linear binary code with help of Mathematica.

*Solution* 2:  
To do this one can use the code generated in example 2 (see fig. 3)



```

In[38]:= B[1] = {0, 0, 1, 1, 1};
B[2] = {0, 1, 0, 1, 0};
B[3] = {0, 1, 1, 0, 1};
B[4] = {1, 0, 0, 1, 0};
B[5] = {1, 0, 1, 0, 1};
B[6] = {1, 1, 0, 0, 0};
B[7] = {1, 1, 1, 1, 1}; MM = {B[1], B[2], B[3], B[4], B[5], B[6], B[7]}
(* Prepare binary sequences *)

{{0, 0, 1, 1, 1}, {0, 1, 0, 1, 0},
{0, 1, 1, 0, 1}, {1, 0, 0, 1, 0},
{1, 0, 1, 0, 1}, {1, 1, 0, 0, 0},
{1, 1, 1, 1, 1}}

In[41]:= For[j = 1, j <= 6, j++,
  For[k = j + 1, k <= 7, k++,
    BB = BitXor[B[j], B[k]];
    Print["j=", j, " k=", k, " ", MemberQ[MM, BB]]]]
(* Organize a loop for checking necessary properties *)

```

Figure 3. MATHEMATICA code for solution of the exercise 2

After executing this code we see the same results as on fig.2.

**Definition 2.** Linear code is called  $[n, k]$  binary code if the set  $V$  in def. 1 includes  $n$ -dimensional binary vectors and

the number of these vectors is equal to  $2^k$   
**Definition 3.** The *weight* of a vector is the number of ones it contains.

For example, the weight of 1101001 is equal to 4.

**Definition 4.** The minimum weight  $d$  of a code is the smallest weight of any nonzero (not all zeros) code word in the code. By term "code word" we mean a vector from linear code space. The minimum weight can be computed with help of Mathematica and similar programs. When the minimum weight is known, it is listed as the third parameter of the code. I.e. we have a

deal with  $[n, k, d]$  -code, which can correct  $t = \left\lfloor \frac{d-1}{2} \right\rfloor$  errors. Hence the minimum weight  $d$  tells us how good the code is.

**Definition 5.** Let introduce a *field* as set of mathematical objects, which can add, subtract, multiply and divide [6].

The simplest field consists of two elements – 0 and 1 with two base operations addition modulo 2  $\oplus$  (see above) and multiplication modulo 2  $\otimes$  (see below):

$$0 \otimes 0 = 0; 0 \otimes 1 = 0; 1 \otimes 0 = 0; 1 \otimes 1 = 1.$$

**Definition 6.** Alphabet, consisting of two symbols 0 and 1 with two operations  $\oplus$  and  $\otimes$  is called *Galois field* of two elements (binary Galois field) and is denoted as  $GF(2)$ .

One can represent an  $[n, k]$ -code by exactly  $k$  linear independent vectors with elements in  $GF(2)$ . These  $k$  vectors are written as the rows of a  $k \times n$  matrix, which is called a *generator matrix*. The generator matrix makes it possible to "generate" all of the vectors in the code by taking all possible linear combinations of these matrix rows, hence its name.

### 3. HAMMING CODE

Let's consider the generator

$$\text{matrix } G = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

,corresponding to  $[7,4,3]$  code (see [7]). This code consists of  $2^4=16$  code words (possible messages).

To encode information word, it is sufficient to find corresponding linear combination of  $G$  lines. In other words, if  $a = a_1 \dots a_k$  - information word, then corresponding code word is equal to  $c = aG$ . Let

$a = (a_1, a_2, a_3)$  then multiplication by  $G^T$  is performed according to the rule:

$$c = (a_1, a_2, a_3, a_4) \times \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} =$$

$$= [a_1 \oplus a_2 \oplus a_3 \quad a_1 \oplus a_2 \oplus a_4 \quad a_2 \quad a_1 \oplus a_3 \oplus a_4 \quad a_3 \quad a_4 \quad a_1].$$

So when the code word is received, assuming no errors have occurred, we can read the original message from the 7<sup>th</sup>, 3<sup>rd</sup>, 5<sup>th</sup>, and 6<sup>th</sup> positions (see yellow highlighting in (1)).

**Example 3.**

Let's find code word for information word  $a = (1100)$ , using (1). Substituting  $a_1$  by 1,  $a_2$  by 1,  $a_3$  and  $a_4$  by 0 and we get  $c = (1 \oplus 1 \oplus 0 \quad 1 \oplus 1 \oplus 0 \quad 1 \quad 1 \oplus 0 \oplus 0 \quad 0 \quad 0 \quad 1) = (0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1)$ .

**Exercise 3**

Find code word for information word  $a = (1100)$  and generator matrix

$$G_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

**Solution**

3:

Let's generate multiplication rule similar to (1) first:

$$c = (a_1, a_2, a_3, a_4) \cdot \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} =$$

$$= [a_1 \quad a_2 \quad a_3 \quad a_4 \quad a_1 \oplus a_2 \oplus a_4 \quad a_1 \oplus a_3 \oplus a_4 \quad a_2 \oplus a_3 \oplus a_4].$$

Then we substitute  $a_1$  by 1,  $a_2$  by 1,  $a_3$  and  $a_4$  by 0 and get

$$c = (1 \quad 1 \quad 0 \quad 0 \quad 1 \oplus 1 \oplus 0 \quad 1 \oplus 0 \oplus 0 \quad 1 \oplus 0 \oplus 0) = (1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1).$$

**Definition 7.** If  $C$  is an  $[n, k, d]$  code, one can define  $C^\perp = \{v \in V \mid v \cdot c = 0 \text{ for all } c \in C\}$ , where " $\cdot$ "

means the dot product (mod 2).  $C^\perp$  is called *dual* of the code  $C$ .

**Definition 8.** A *parity check matrix* of a code  $C$  is an  $(n - k) \times n$  matrix where the rows are linearly independent vectors such that the dot product (mod 2) of each row with any vector in the code is equal to 0.

$$\bar{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & p_{00} & p_{01} & p_{02} \\ 0 & 1 & 0 & 0 & p_{10} & p_{11} & p_{12} \\ 0 & 0 & 1 & 0 & p_{20} & p_{21} & p_{22} \\ 0 & 0 & 0 & 1 & p_{30} & p_{31} & p_{32} \end{bmatrix}.$$

A parity check matrix can be easily recovered using

$$P = \begin{bmatrix} p_{00} & p_{01} & p_{02} \\ p_{10} & p_{11} & p_{12} \\ p_{20} & p_{21} & p_{22} \\ p_{30} & p_{31} & p_{32} \end{bmatrix} \text{ (see [8]).}$$

It has a special (canonical) form

$$\bar{H} = [P \mid E]$$

where  $P^T$  is transposed matrix  $P$  and  $E$  is unit matrix, which has ones on the main diagonal and zeros in the other places. E.g., for the generator  $G_1$  we have

$$P = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \text{ so } P^{-1} = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix} \text{ and}$$

$$\bar{H} = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}. \quad (3)$$

If parity check matrix has no canonical form, then one can reduce it to (2) by elementary transformations over its rows modulo 2.

Example 4

Let's find parity check matrix for generator matrix  $G$  (roman numerals indicate the rows of the matrix):

$$G = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \sim (I \leftrightarrow III) \sim \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \sim (II \leftrightarrow IV) \sim$$

$$\sim \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \sim (I \oplus III, I \oplus IV) \sim \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \end{bmatrix} \sim (III \leftrightarrow IV) \sim$$

$$\sim \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \sim (II \oplus III, II \oplus IV) \sim \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \sim (IV \oplus III) \sim$$

$$\sim \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \sim (IV \oplus II, IV \oplus I) \sim \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \sim (IV + III) \sim$$

$$\sim \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

i.e.  $P = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$  and  $P^T = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix}$ .

. Hence the parity check matrix for the Hamming code

$$G \text{ is } H = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

**Exercise 4.** Find parity check matrix for the matrix  $G_2 = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}$

**Solution 4**

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix} \sim (I \leftrightarrow IV) \sim \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} = [E | P].$$

So,  $P^T = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix}$  and  $H_2 = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$ .

Example 5

Let's check, that  $h_i \times g_j = 0 \pmod 2$  where  $h_i$  is the row of  $H$  and  $g_j$  is the row of  $G$  for  $i = j = 1$ .

We can highlight the nonzero components, being at the same positions of  $h_1$  and  $g_1$ :

$g_1 = (1 \ 1 \ 0 \ 1 \ 0 \ 0 \ 1)$ ,  $h_1 = (0 \ 1 \ 1 \ 1 \ 0 \ 0)$ . Dot product  $h_1 \times g_1 = 1 \oplus 1 = 0 \pmod 2$ .

Exercise 5

Check, that  $h_i \times g_j = 0 \pmod 2$  for  $i=2, j=3$ , where  $g_j$  is a row of matrix  $G_2$  and  $h_j$  is a row of matrix

$H_2$ .

Solution 5

There are no nonzero components being at the same positions of  $h_2$  and  $g_3$ :

$h_2 = (0 \ 1 \ 1 \ 1 \ 0 \ 1 \ 0)$ ,  $g_3 = (1 \ 0 \ 0 \ 0 \ 1 \ 0 \ 1)$ . Hence,  $h_2 \times g_3 = 0 \pmod 2$ .

## 4. SYNDROME DECODING

**Definition 9.** If we take any vector  $v \in V$  and multiply  $v^t$  (where  $v^t$  is the transposition of the vector  $v$ ) on the left by the parity check matrix then we get a vector  $s$  called the *syndrome* of  $v$ . It indicates whether there are any errors in the code word.

Example 6

Let's compute the syndrome  $s$  of the vector  $c = [0 \ 0 \ 1 \ 1 \ 1 \ 0 \ 1]$  which was received by Bob, using Hamming parity check matrix with help of Mathematica.

$$H \cdot \hat{c}^t = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \oplus 1 \oplus 1 \\ 1 \oplus 1 \\ 1 \oplus 1 \oplus 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}.$$

To check this result we will organize a simple loop (Fig.4) Exercise 6

Compute the syndromes of the vector  $b = [0 \ 1 \ 1 \ 0 \ 1 \ 0 \ 1]$

a. By Hand

b. With help of Mathematica

```

File Edit Insert Format Cell Graphics Evaluation Palettes Window Help
H = ( 0 0 0 1 1 1 1
      0 1 1 0 0 1 1
      1 0 1 0 1 0 1 ) (* Hamming parity check matrix *)

Out[21]= {{0, 0, 0, 1, 1, 1, 1}, {0, 1, 1, 0, 0, 1, 1}, {1, 0, 1, 0, 1, 0, 1}}

e = {0, 0, 1, 1, 1, 0, 1} (* vector c *)

Out[22]= {0, 0, 1, 1, 1, 0, 1}

In[108]:= c = {0, 0, 0, 0, 0, 0, 0}; cc = {0, 0, 0};
For[i = 1, i <= 3, i++, d = 0;
(* loops for calculating of dot product H.c *)
For[j = 1, j <= 7, j++,
c[[j]] = H[[i, j]] * e[[j]],
d = BitXor[d, c[[j]]]];
cc[[i]] = d]

Out[110]= {1, 0, 1}

```

Figure 4. Calculating a syndrome

Solution 6

$$\begin{aligned}
 \text{a. } H \cdot b^t &= \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} \\
 &= \begin{bmatrix} 1 \oplus 1 \\ 1 \oplus 1 \oplus 1 \\ 1 \oplus 1 \oplus 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}
 \end{aligned}$$

b. Using the same code, we have

```

In[124]:= b = {0, 1, 1, 0, 1, 0, 1}; k = {0, 0, 0, 0, 0, 0, 0};
For[i = 1, i <= 3, i++, d = 0;
(* loops for calculating of dot product H.c *)
For[j = 1, j <= 7, j++,
k[[j]] = H[[i, j]] * b[[j]],
d = BitXor[d, k[[j]]]];
cc[[i]] = d]

Out[126]= {0, 1, 1}

```

Figure 5. Calculating a syndrome

Suppose that initial message was  $c = [0011001]$ , i.e.

there is the single error at the 5<sup>th</sup> position. One can represent  $c$  as a sum  $c = c + e$ , where  $e = [0000100]$ . Then the result is:

$$\begin{aligned}
 H \cdot \hat{c}^t &= H \cdot (c + e) \\
 &= H \cdot c^t + H \cdot e^t = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + H \cdot e^t = H \cdot e^t,
 \end{aligned}$$

because the dot product (mod 2) of a code phrase with each row of the parity check matrix is 0. So the syndrome of the received message is the same as that of its corresponding error vector.

## 5. MCELIECE CRYPTOSYSTEM

Let  $C$  be an  $[n, k]$  linear code with a fast decoding algorithm that can correct  $t$  or fewer errors. Let  $G$  be a generator matrix for  $C$ . To create the disguise, let  $S$  be a  $k \times k$  invertible matrix (the scrambler) and let  $P$  be an  $n \times n$  permutation matrix (i.e., having a single 1 in each row and column and 0's everywhere else). The matrix

$$G' = SG$$

is made public while  $S$ ,  $G$  and  $P$  are kept secret by Bob. For Alice to send a message to Bob, she blocks her message into binary vectors of length  $k$ . If  $x$  is one such block, she randomly constructs a binary  $n$ -vector of weight  $t$  (that is, she randomly places  $t$  1's in a zero vector of length  $n$ ), call it  $e$  and then sends to Bob the vector

$$y = xG' + e.$$

Eve, upon intercepting this message, would have to find the nearest codeword to  $y$  of the code

generated by  $G'$ . This would involve calculating the syndrome of  $y$  and comparing it to the

syndromes of all the error vectors of weight  $t$ .

As there are  $\binom{n}{t}$  of these error vectors, good choices

of  $n$  and  $t$  will make this computation infeasible. Bob, on

the other hand, would calculate<sup>1</sup>

$$y' = yP^{-1} = (xG' + e)P^{-1} = xSG + eP^{-1} = xSG + e', \quad (4)$$

where  $e'$  is a vector of weight  $t$  (since  $P^{-1}$  is also a permutation matrix). Bob now applies the fast decoding algorithm to strip off the error vector  $e'$  and get the code word  $(xS)G$ . The vector  $xS$  can now be obtained by

multiplying by  $G^{-1}$  on the right. Finally, Bob gets  $x$  by multiplying  $xS$  on the right by  $S^{-1}$ .

Example 7.

Suppose that Alice wishes send the message  $x = (1101)$ , using

$$G = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \quad \text{as Hamming generator matrix and vector } e \text{ with weight 1.}$$

Assuming  $e = (0000010)$ .

$$\text{On other hand Bob chooses the scrambler matrix } S = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

<sup>1</sup> Inverse matrix  $A^{-1}$  for given square matrix  $A$  is nondegenerate matrix, satisfying  $A^{-1}A = A A^{-1} = E$  (unit matrix). [9]



and permutation matrix  $P = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$ .

To calculate public generator matrix we will use Mathematica (see fig.5).

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}; G = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}; S = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix};$$

```
In[158]:= R1 = Mod[S.G.P, 2]; MatrixForm[R1]
Out[158]/MatrixForm=

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}$$

```

Figure 6. Calculating a public generator matrix in Mathematica

The result is matrix  $G' = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}$ .

Next step is computing  $y = xG' + e$ :

```
In[161]:= e = {0, 0, 0, 0, 0, 1, 0}; x = {1, 1, 0, 1};
y = Mod[Mod[x.R1, 2] + e, 2]
```

$y = (1 \ 1 \ 0 \ 1 \ 1 \ 1 \ 0)$  is a coding message, which Alice send to Bob.

Upon receiving  $y$ , Bob first compute  $P^{-1}$  and multiply it on the left by  $y$ . Applying Mathematica, we get  $y' = yP^{-1} = (1 \ 1 \ 0 \ 1 \ 0 \ 1 \ 1)$  (see fig.6).

```
In[163]:= PP = MatrixForm[Inverse[P]]
Out[163]/MatrixForm=

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

In[162]:= y1 = y.PP
Out[162]= {1, 1, 0, 1, 0, 1, 1}
```

Figure 7. Calculating  $P^{-1}$  and  $y'$  in Mathematica

Then we calculate a syndrome of  $y'$ , using definition 9 (see also example 6). Since we have parity check matrix for  $G'$ ,

$$H' = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

(see exercise 4), one can easily compute  $s' = y' * (H')^T$  with help of Mathematica (see below).

```
H1T = Transpose[H1]; Mod[y1.H1T, 2] (* syndrome of y' *)
Out[217]= {0, 1, 0}
```

To identify this result we should generate a table of syndromes. It is not difficult using Mathematica:

```
e = {0, 0, 0, 0, 0, 0, 0};
For[i = 1, i <= 7, i++, f = e;
f[[i]] = e[[i]] + 1; s = f.H1T; Print["e=", f, " s=", s]
] (* generating a table of syndromes *)

e={1, 0, 0, 0, 0, 0, 0} s={1, 0, 1}
e={0, 1, 0, 0, 0, 0, 0} s={1, 1, 0}
e={0, 0, 1, 0, 0, 0, 0} s={0, 1, 1}
e={0, 0, 0, 1, 0, 0, 0} s={1, 1, 1}
e={0, 0, 0, 0, 1, 0, 0} s={1, 0, 0}
e={0, 0, 0, 0, 0, 1, 0} s={0, 1, 0}
e={0, 0, 0, 0, 0, 0, 1} s={0, 0, 1}
```

Figure 8. The table of syndromes

Hence error occurs in 6<sup>th</sup> position. Now Bob can correct the word  $y'$  and get a word

$y'' = y' - e_4 = (1 \ 1 \ 0 \ 1 \ 0 \ 0 \ 1)$ . Because of special choice of  $G$  (see comments after (1)) he reads the original message from the 7<sup>th</sup>, 3<sup>rd</sup>, 5<sup>th</sup>, and 6<sup>th</sup> positions, i.e.  $xS = (1 \ 0 \ 0 \ 0)$ . The last step is obtaining  $x$  through multiplying by matrix  $S^{-1}$ :

```
In[194]:= MatrixForm[Mod[Inverse[S], 2]]
Out[194]/MatrixForm=

$$\begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

```

Finally,  $x = (1 \ 0 \ 0 \ 0) \times \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix} = (1 \ 1 \ 0 \ 1)$ .

As we see, the result is correct.

#### Exercise 7

Reconstruct encryption- decryption process for the message  $x = (1 \ 0 \ 1 \ 1)$ , assuming  $e = (0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0)$  with the same generator matrix and another scrambler matrix

$$S = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix}$$

and

permutation matrix  $P =$

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Solution 7

1. Calculating public generator matrix

$$P1 = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}; S1 = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}; R2 = \text{Mod}[S1.G.P1, 2];$$

MatrixForm[R2]

```
//MatrixForm=
```

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}$$

2. Computing coding message  $y = xG' + e$ :

```
In[219]:= x = {1, 0, 1, 1}; e5 = {0, 0, 0, 0, 1, 0, 0}; y = Mod[x.R2 + e5, 2]
Out[219]:= {1, 0, 1, 1, 1, 1, 0, 1}
```

3. Calculate  $y' = yP^{-1}$

```
In[220]:= P1I = Inverse[P1]; y1 = y.P1I
Out[220]:= {0, 1, 1, 1, 1, 1, 0, 1}
```

4. Computing syndrome of  $y'$ , using the same parity check matrix H1:

```
Mod[y1.H1T, 2] (* syndrome *)
```

```
Out[221]:= {0, 0, 1}
```

and find correspondent error at the table of syndromes (see fig. 7). Here we see, that the

error occurs in 8<sup>th</sup> position.

5. Correcting  $y'$  and recover  $y''$ .

Similar to the example, we get a word

$y'' = y' - e8 = (0 \ 1 \ 1 \ 1 \ 0 \ 0)$ . Because of special choice of G (see comments after (1)) he read the original

message from the 7<sup>th</sup>, 3<sup>rd</sup>, 5<sup>th</sup>, and 6<sup>th</sup> positions, i.e.  $xS = (0 \ 1 \ 1 \ 0)$ .

6. Obtaining  $x$  thro multiplying previous vector by the matrix  $S^{-1}$ :

```
In[223]:= x1 = {0, 1, 1, 0}; S1I = Mod[Inverse[S1], 2]; Mod[x1.S1I, 2]
```

```
Out[223]:= {1, 0, 1, 1}
```

This is the message, sent by Alice.

## 6. CONCLUSION

This research paper about the McEliece Cryptosystem gave me an idea of how my data could be better secured in the future. Generating the key and authentication process was simple in form but complicated in decoding. However, I was successfully able to recreate the encryption system using Mathematica and delineate the methods of encryption and decryption.

In addition to learning about the mathematics behind the McEliece Cryptosystem, the process of exploration not only taught me how to use Mathematica but also how to structure my research. Exploring a way for us to keep our data safe from hackers has been a very valuable experience for me, as data security is one of the hottest recent topics. Small efforts made by individuals such as myself can protect much of our personal data, rather than relying on other services to protect our data for us.

The topic of cryptography has always been the matter of how information could be securely stored secretly to prevent hackers from accessing the information and possibly causing harm. The method of encryption that one puts on his/her information is quite an important matter as it decides how easily the hackers are able to access the information at all. The fast developing information technology industry has brought up increasingly tough hackers and encryption software, bringing concerns to the public. The fast developing hacking industry has also caused developers to come up with more secure and complicated encryption systems.

In order to make the decryption more complicated, longer keys must be implemented. The dawn of quantum computing will make this especially important. When quantum computers directly employ the quantum mechanical phenomena to perform the operation of data, with magnitudes greater processing speed, they will be able to break many current encryption systems with simple brute force attacks. Coming up with ways to stay safe from these brute force attacks is a pressing concern in the encryption community, and I would love to be able to contribute to helping solve this problem in the future.

In the meantime, I'd like to further study other well-known encryption systems like AES, RSA, elliptic curve, Triple DES, and Blowfish. Each of these systems employs very unique methods of encryption and decryption, with pros and cons for each one. With the variety, I'd like to learn about all the different approaches so that in the future, I can better understand what makes a successful encryption system. If possible, I would like to be part of developing a unique encryption system of the future.

## REFERENCES

- Wang, Yongge (2015). *Random Linear Code Based Public Key Encryption Scheme RLCE*, URL: <https://eprint.iacr.org/2015/298>
- Bernstein, Daniel J., Tanja Lange, Christiane Peters (2008), *Attacking and Defending the McEliece Cryptosystem*, URL: <https://cr.yp.to/codes/mceliece-20080807.pdf>
- Peters, Christiane. *Information set-decoding for linear codes over  $F^2$* , URL: <https://eprint.iacr.org/2009/589.pdf>
- Cook, Tim. *A Message to Our Customers*. URL: <http://www.apple.com/customer-letter/>
- S.Au, C. Eubanks-Turner, J. Everson. *The McEliece Cryptosystem*. URL: <http://www.math.unl.edu/~s-jeverso2/McElieceProject.pdf>
- Yunghsiang S. Han. *Introduction to Finite Fields* URL: <http://web.ntpu.edu.tw/~yshan/algebra.pdf>
- Dipperstein, Michael. *Hamming(7,4) Code Discussion and Implementation* URL: [http://michael.dipperstein.com/hamming/Hamming\(7,4\).](http://michael.dipperstein.com/hamming/Hamming(7,4).) URL: [https://en.wikipedia.org/wiki/Hamming\(7,4\)\\_Matrix\\_Algebra\\_Tutorial](https://en.wikipedia.org/wiki/Hamming(7,4)_Matrix_Algebra_Tutorial) URL: <http://stattrek.com/tutorials/matrix-algebra-tutorial.aspx>
- McEliece Cryptosystem URL: [https://en.wikipedia.org/wiki/McEliece\\_cryptosystem](https://en.wikipedia.org/wiki/McEliece_cryptosystem)
- McEliece\_cryptosystem Hands-on Start to Mathematica URL: <https://www.wolfram.com/broadcast/screencasts/handsonstart/>

## APPENDIX

### Mathematica Encryption Code

```
For [j = 1, j ≤ 6, j++,
For [k = j + 1, k ≤ 7, k++, BB = BitXor[B[j],
B[k]];

Print[SubsetQ[M, BB]]];

ClearAll[B]

B[1] = {1, 0, 0, 1, 1}; B[2] = {0, 1, 0,
1, 0}; B[3] = {1, 1, 0, 0, 1}; B[4] = {0,
0, 1, 0, 1}; B[5] = {1, 0, 1, 1, 0}; B[6]
= {0, 1, 1, 1, 1};

B[7] = {1, 1, 1, 0, 0}; (* Prepare binary
sequences *)

M = {B[1], B[2], B[3], B[4], B[5], B[6], B[7]} (*
Prepare set of sequences *)

{{1, 0, 0, 1, 1}, {0, 1, 0, 1, 0}, {1, 1, 0, 0,
1},
{0, 0, 1, 0, 1}, {1, 0, 1, 1, 0}, {0, 1, 1, 1, 1},
{1, 1, 1, 0, 0}}

For [j = 1, j ≤ 6, j++,

For [k = j + 1, k ≤ 7, k
++,

BB = BitXor[B[j], B[k]];
Print["j=", j, " k=", k, " ", MemberQ[M, BB]]]
(* Organize a loop for checking necessary
properties *)

j=1 k=2
True

j=1 k=3 True BitXor[B[1], B[2]]
j=1 k=4 True {1, 1, 0, 0, 1}
j=1 k=5 True
j=1 k=6 True MemberQ[M, {1, 1, 0, 0, 1}]
j=1 k=7 True True
j=2 k=3 True
j=2 k=4 True B[1] = {0, 0, 1, 1, 1};
j=2 k=5 True B[2] = {0, 1, 0, 1, 0};
j=2 k=6 True B[3] = {0, 1, 1, 0, 1};
j=2 k=7 True B[4] = {1, 0, 0, 1, 0};
j=3 k=4 True B[5] = {1, 0, 1, 0, 1};
j=3 k=5 True B[6] = {1, 1, 0, 0, 0};
j=3 k=6 True
j=3 k=7 True
j=4 k=5 True
j=4 k=6 True
j=4 k=7 True
j=5 k=6 True
j=5 k=7 True
j=6 k=7 True

B[7] = {1, 1, 1, 1, 1}; MM = {B[1], B[2],
B[3], B[4], B[5], B[6], B[7]} (* Prepare
binary sequences *)
{{0, 0, 1, 1, 1}, {0, 1, 0, 1, 0},
{0, 1, 1, 0, 1}, {1, 0, 0, 1, 0},
{1, 0, 1, 0, 1}, {1, 1, 0, 0, 0},
{1, 1, 1, 1, 1}}

For [j = 1, j ≤ 6, j++,
```



```

BB = BitXor[B[j], B[k]];
Print["j=", j, " k=", k, " ",
MemberQ[MM, BB]]]
(* Organize a loop for checking
necessary properties *)

```

```

j=1 k=2 True
j=1 k=3 True
j=1 k=4 True
j=1 k=5 True
j=1 k=6 True
j=1 k=7 True
j=2 k=3 True
j=2 k=4 True
j=2 k=5 True
j=2 k=6 True
j=2 k=7 True
j=3 k=4 True
j=3 k=5 True
j=3 k=6 True
j=3 k=7 True
j=4 k=5 True
j=4 k=6 True
j=4 k=7 True
j=5 k=6 True
j=5 k=7 True
j=6 k=7 True

```

## MATHEMATICA DECRYPTION CODE

$$H = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

```

(* Hamming parity check matrix *)
{{0, 0, 0, 1, 1, 1, 1}, {0, 1, 1, 0, 0, 1, 1},
1, 1}, {1, 0, 1, 0, 1, 0, 1}}

```

```

e = {0, 0, 1, 1, 1, 0, 1} (* vector c *)

```

```

{0, 0, 1, 1, 1, 0, 1}

```

```

c = {0, 0, 0, 0, 0, 0, 0}; cc = {0, 0,
0};

```

```

For [i = 1, i ≤ 3, i++, d = 0;
(* loops for calculating of dot product
H.c *)

```

```

For [j = 1, j ≤ 7, j++;
c[[j]] = H[[i, j]] * e[[j]],

```

```

d = BitXor[d, c[[j]]] ];
cc[[i]] = d]
cc (* result *)
{1, 0, 1}

```

```

b = {0, 1, 1, 0, 1, 0, 1};
k = {0, 0, 0, 0, 0, 0, 0};
For [i = 1, i ≤ 3, i++, d = 0;
(* loops for calculating of dot product
H.c *)

```

```

For [j = 1, j ≤ 7, j++;

```

```

k[[j]] = H[[i, j]] * b[[j]],

```

```

d = BitXor[d, k[[j]]] ]; cc[[i]] = d]

```

```

cc

```

```

{0, 1, 1}

```

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix};$$

$$G = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix};$$

$$S = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix};$$

```

R1 = Mod[S.G.P, 2]; MatrixForm[R1]

```

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}$$

```

e = {0, 0, 0, 0, 0, 1, 0};

```

```

x = {1, 1, 0, 1};

```

```

y = Mod[Mod[x.R1, 2] + e, 2] {1, 1, 0, 1,
1, 1, 0}

```

```

PP = MatrixForm[Inverse[P]]

```

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

```

y1 = y.PP
{1, 1, 0, 1, 0, 1, 1}

```

```

H1T = Transpose[H1]; Mod[y1.H1T, 2] (* syndrome of y' *)
{0, 1, 0}

```

```

e = {0, 0, 0, 0, 0, 0, 0};
For [i = 1, i ≤ 7, i++, f = e;
f[[i]] = e[[i]] + 1; s = f.H1T; Print["e=", f, " s=", s]
] (* generating a table of syndromes *)

```

```

e={1, 0, 0, 0, 0, 0, 0} s={1, 0, 1}
e={0, 1, 0, 0, 0, 0, 0} s={1, 1, 0}
e={0, 0, 1, 0, 0, 0, 0} s={0, 1, 1}
e={0, 0, 0, 1, 0, 0, 0} s={1, 1, 1}
e={0, 0, 0, 0, 1, 0, 0} s={1, 0, 0}
e={0, 0, 0, 0, 0, 1, 0} s={0, 1, 0}
e={0, 0, 0, 0, 0, 0, 1} s={0, 0, 1}

```

```

e = {0, 0, 0, 0, 0, 0, 0};

```

```

For [i = 1, i ≤ 7, i++, f = e;
f[[i]] = e[[i]] + 1; s = f.H1T; Print["e=", f, " s=", s]
] (* generating a table of syndromes *)

```

```

e={1, 0, 0, 0, 0, 0, 0} s={1, 0, 1}
e={0, 1, 0, 0, 0, 0, 0} s={1, 1, 0}
e={0, 0, 1, 0, 0, 0, 0} s={0, 1, 1}
e={0, 0, 0, 1, 0, 0, 0} s={1, 1, 1}
e={0, 0, 0, 0, 1, 0, 0} s={1, 0, 0}
e={0, 0, 0, 0, 0, 1, 0} s={0, 1, 0}
e={0, 0, 0, 0, 0, 0, 1} s={0, 0, 1}

```

```

MatrixForm[Mod[Inverse[S], 2]] (* S^{-1} *)

```

$$\begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

$$P1 = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix};$$

$$S1 = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix};$$

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}$$

```

x = {1, 0, 1, 1}; e5 = {0, 0, 0, 0, 1,
0, 0}; y = Mod[x.R2 + e5, 2] {1, 0, 1,
1, 1, 0, 1}

```

```

P1I = Inverse[P1]; y1 = y.P1I

```

```

{0, 1, 1, 1, 1, 0, 1}

```

*McEllice#2.nb*

2

```

Mod[y1.H1T, 2      syndrome *)

```

```

{0, 0, 1}

```

```

x1 = {0, 1, 1, 0}; S1I =

```

```

Mod[Inverse[S1], 2]; Mod[x1.S1I, 2]

```

```

{1, 0, 1, 1}

```