

POLYTECHNIC UNIVERSITY  
Department of Computer and Information Science

The Hill Cipher

K. Ming Leung

**Abstract:** The Hill cipher in cryptography is used to illustrate the application of matrices defined over a finite field, and the handling of characters and strings in computer programs.

Directory

- [Table of Contents](#)
- [Begin Article](#)

Copyright © 2000 [mleung@poly.edu](mailto:mleung@poly.edu)  
Last Revision Date: November 16, 2004

# Table of Contents

1. Introduction
2. Arithmetic over a finite field
3. Example of a Finite Field
4. Character Set and Strings
5. The  $m = 2$  Hill Cipher

## 1. Introduction

Cryptography, to most people, is concerned with keeping communications private. Indeed, the protection of sensitive communications has been the emphasis of cryptography throughout much of its history. As we will see, however, this is only one part of today's cryptography.

Encryption is the transformation of data into some unreadable form. Its purpose is to ensure privacy by keeping the information hidden from anyone for whom it is not intended, even those who can see the encrypted data. Decryption is the reverse of encryption ; it is the transformation of encrypted data back into some intelligible form.

Encryption and decryption require the use of some secret information, usually referred to as a key. Depending on the encryption mechanism used, the same key might be used for both encryption and decryption, while for other mechanisms, the keys used for encryption and decryption might be different.

But today's cryptography is more than secret writing, more than encryption and decryption. Authentication is as fundamental a part of our lives as privacy. We use authentication though out our ev-

everyday life, when we sign our name to some document for instance, and as we move to a world where our decisions and agreements are communicated electronically, we need to replicate these procedures.

Cryptography provides mechanisms for such procedures. A digital signature binds a document to the possessor of a particular key, while a digital timestamp binds a document to its creation at a particular time. These cryptographic mechanisms can be used for example to control access to a shared disk drive, a high security installation or to a pay-per-view TV channel.

While modern cryptography is growing increasingly diverse, cryptography is fundamentally based on problems that are difficult to solve. A problem may be difficult because its solution requires some secret knowledge, such as decrypting an encrypted message or signing some digital document, or the problem may be hard because it is intrinsically difficult to complete, such as finding a message which produces a given hash value.

We illustrate here some simple ideas concerning encryption and decryption using the cryptographic technique known as the Hill cipher. Hill cipher involves the use of  $n \times n$  matrices defined over a finite

field. We will also take the opportunity to illustrate how characters and strings can be handled in (Matlab) programs.

## 2. Arithmetic over a finite field

A field is a set of numbers with an addition operation and a multiplication operation defined so that the set of numbers are closed under these operations. That means that the result of adding or multiplying any two numbers in the set has to be in the set as well.

There has to be an identity element for addition (referred to as 0), such that adding 0 to any element of the set gives exactly the same element.

And there has to be an identity element for multiplication (referred to as 1), such that multiplying any element by the unit element, 1, gives back the same element.

Some other familiar properties are also required: there are additive and multiplicative inverses for every number (except that zero has no multiplicative inverse); and the commutative, associative, and distributive laws are obeyed in exactly the same ways as in ordinary

arithmetic.

Examples of a field is the set of real numbers with ordinary operations of addition and multiplication. Other examples include the set of complex numbers and rational numbers with ordinary addition and multiplication operations. However, all these examples of fields have an infinite number of elements. A field with a finite number of elements is called a finite field (also known as a Galois field). It has the interesting feature that arithmetic could be done exactly by finite machines, such as computers. Finite-field arithmetic has important applications such as error-correcting codes and cryptography.

### 3. Example of a Finite Field

If  $p$  is a prime number, then the set of integers  $0, 1, 2, \dots, p-1$  with the usual addition and multiplication constitutes a finite field, if all arithmetics are carried out modulo  $p$ . The order of a field is defined as the number of elements in the set. This field is denoted by  $GF(p)$ , which stands for Galois field of order  $p$ . Numbers are considered identical to each other if they give the same remainders after dividing

by  $p$ . Closure of the set under these operations is therefore evident. Given any integer  $m$ , its integer remainder after dividing by  $p$  is given in C and C++ by  $m\%p$ , and by `mod(m,p)` in Matlab.

As an example, let us take  $p = 5$  and consider the set  $\mathbf{Z}_5$  which has elements  $0, 1, 2, 3, 4$ . The addition table is

+	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	5
2	2	3	4	5	6
3	3	4	5	6	7
4	4	5	6	7	8

Taking mod 5 of the result gives

+	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

The multiplication table is

$\times$	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	6	8
3	0	3	6	9	12
4	0	4	8	12	16

Taking mod 5 of the result gives



$\times$	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

From the table we see that  $3 \times 2 \equiv 1$  and therefore 3 and 2 are multiplicative inverses of each other. Thus we have

$$1^{-1} = 1, \quad 2^{-1} = 3, \quad 3^{-1} = 2, \quad 4^{-1} = 4.$$

Notice that the last element of the set 4 is its own inverse. This is always true because  $\text{mod}((p-1)^2, p) = \text{mod}(p^2 - 2p + 1, p) = \text{mod}(1, p) = 1$ , that means that  $p-1$  is its own inverse. It is clear that this is true even if  $p$  is not a prime.

If  $p$  is not a prime number, then the set of integers  $0, 1, 2, \dots, p-1$  with the usual addition and multiplication carried out modulo  $p$  does not constitute a finite field. To illustrate we consider the case  $p = 6$ . The multiplication table is:

$\times$	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	1	2	3	4	5
2	0	2	4	6	8	10
3	0	3	6	9	12	15
4	0	4	8	12	16	20
5	0	5	10	15	20	25

Taking mod 6 of the result gives

$\times$	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	1	2	3	4	5
2	0	2	4	0	2	4
3	0	3	0	3	0	3
4	0	4	2	0	4	2
5	0	5	4	3	2	1

We see that 2, 3 and 4 have no multiplicative inverses. Thus we

do not have a finite field.

## 4. Character Set and Strings

Computers use the ASCII character set to store basic text. The character set uses seven of the eight bits in a byte to encode 128 characters. The first 32 characters are non-printing control characters, such as tab, backspace and end-of-line. The 128th character is another non-printing character representing the delete key. In between these control characters are 95 printable characters, including a space, 10 digits, 26 lowercase letters, 26 uppercase letters and 32 punctuation marks.

Matlab can easily display all the printable characters, in the order determined by their ASCII encoding. Start with

```
x = reshape(32:127,32,3)'
```

This produces a 3-by-32 matrix.

```
x =
```

```
    32    33    34    ...    61    62    63
```

64	65	66	...	93	94	95
96	97	98	...	125	126	127

The `char` function converts numbers to characters. The statement

```
c = char(x)
```

produces

```
c =  
!"#$%&'()*+,-./0123456789:;<=>?  
@ABCDEFGHIJKLMNPOQRSTUVWXYZ[\]^_  
'abcdefghijklmnopqrstuvwxyz{|}~
```

Actually the last element of `x` is 127, which corresponds to the non-printing delete character. We have not try to display it here since it may appear differently depending on your computer.

The first character in `c`, `char(32)`, is the blank character. The last printable character in `c`, `char(126)`, is the tilde. The characters representing digits are in the first line of `c`. In fact

```
d = char(48:57)
```

displays a ten-character string

```
d =  
0123456789
```

This string can be converted to the corresponding numerical values with `double` or `real`. The statement

```
double(d) - '0'
```

produces

```
0 1 2 3 4 5 6 7 8
```

Comparing the second and third line of `c`, we see that the ASCII encoding of the lowercase letters is obtained by adding 32 to the ASCII encoding of the uppercase letters. Understanding this encoding allows us to use vector and matrix operations in Matlab to manipulate text.

Our encryption technique involves modular arithmetic. All the quantities involved are integers and the result of any arithmetic operation is reduced by taking the remainder or modulus with respect to a prime number,  $p$ . The Matlab functions `rem(x,y)` and `mod(x,y)` both compute the remainder if  $x$  is divided by  $y$ . They produce the same result if  $x$  and  $y$  have the same sign; the result also has that sign. But if  $x$  and  $y$  have opposite signs, then `rem(x,y)` has the same

sign as  $x$ , while  $\text{mod}(x,y)$  has the same sign as  $y$ . Thus we will be using the mod function here. Here is a table.

```
x = [37 -37 37 -37]';  
y = [10 10 -10 -10]';  
r = [ x y rem(x,y) mod(x,y)]
```

produces

```
37  10  7  7  
-37 10 -7  3  
37 -10  7 -3  
-37 -10 -7 -7
```

## 5. The $m = 2$ Hill Cipher

We have chosen to encrypt text that uses the entire ASCII character set, not just the letters. There are 95 such characters. The next larger prime number is  $p = 97$ , so we represent the  $p$  characters by the integers  $0:p-1$  and do arithmetic mod  $p$ . The characters are encoded two at a time. Each pair of characters is represented by a 2-vector,  $\mathbf{x}$ . For example, suppose the text contains the pair of letters 'TV'. The

ASCII values for this pair of letters are 84 and 86. Subtracting 32 to make the representation start at 0 produces the column vector

$$\mathbf{x} = \begin{pmatrix} 52 \\ 54 \end{pmatrix}$$

The encryption is done with a 2-by-2 matrix-vector multiplication over the integers mod  $p$ .

$$\mathbf{y} = \mathbf{Ax}; \quad \text{mod } p$$

where  $\mathbf{A}$  is the matrix

$$\mathbf{A} = \begin{pmatrix} 71 & 2 \\ 2 & 26 \end{pmatrix}$$

For our example, the product  $\mathbf{Ax}$  is

$$\mathbf{Ax} = \begin{pmatrix} 3800 \\ 1508 \end{pmatrix}$$

If this is reduced mod  $p$  the result is

$$\mathbf{y} = \begin{pmatrix} 17 \\ 53 \end{pmatrix}$$

Converting this back to characters by adding 32 produces '1U'. Now comes the interesting part. Over the integers modulo  $p$ , the matrix  $A$  is its own inverse. That is if

$$\mathbf{y} = \mathbf{A}\mathbf{x}; \quad \text{mod } p$$

then

$$\mathbf{x} = \mathbf{A}\mathbf{y}; \quad \text{mod } p$$

In other words, in arithmetic mod  $p$ ,  $\mathbf{A}^2$  is the identity matrix. You can check this with Matlab:

```
p = 97;  
A = [71 2; 2 26];  
I = mod(A^2,p)
```

produces

```
I =  
    1 0  
    0 1
```



We can also directly find  $\mathbf{A}^{-1}$ . Since  $\mathbf{A}$  is a  $2 \times 2$  matrix, its inverse is given by

$$\mathbf{A}^{-1} = (\det \mathbf{A})^{-1} \begin{bmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{bmatrix},$$

where the determinant of  $\mathbf{A}$  is given by  $\det \mathbf{A} = a_{11}a_{22} - a_{21}a_{12}$ . Again remember that all operations must be carried out in mod  $p$  ( $=97$ ) arithmetic.

For the above  $\mathbf{A}$ , we have  $\det \mathbf{A} = 71 \times 26 - 2 \times 2 = 1842 = 96$ . Since  $96 = 97 - 1$ , and using a result that we proved earlier, we see that  $96^{-1} = 96$ . Therefore

$$\mathbf{A}^{-1} = 96 \begin{bmatrix} 26 & -2 \\ -2 & 71 \end{bmatrix} = \begin{bmatrix} 2496 & -192 \\ -192 & 6816 \end{bmatrix} = \begin{bmatrix} 71 & 2 \\ 2 & 26 \end{bmatrix}.$$

Thus the inverse of  $\mathbf{A}$  is indeed  $\mathbf{A}$  itself.

The fact that  $\mathbf{A}$  is its own inverse means that the encryption process is its own inverse. The same function can be used to both encrypt and decrypt a message.

The following Matlab function program named `crypto97.m` is an implementation of the above  $m = 2$  Hill cipher. The function begins

with a preamble.

```
function y = crypto97(x)
% CRYPTO Cryptography example.
% y = crypto97(x) converts an ASCII text string into another
% coded string. The function is its own inverse, so
% crypto97(crypto97(x)) gives x back.
```

A comment precedes the statement that assigns the prime  $p$ .

```
% Use a two-character Hill cipher with arithmetic
% modulo 97, a prime.
p = 97;
```

Characters with ASCII values of 169 and 174 are chosen to expand the character set from 95 to 97.

```
% Choose two characters above ASCII 128 to expand set from 95 to 97
c1 = char(169);
c2 = char(174);
x(x==c1) = 127;
x(x==c2) = 128;
```

The printable characters are then converted to integers mod  $p$ .

```
% Convert printable ASCII text to integers mod p.  
x = mod(real(x-space),p);
```

Prepare for the matrix-vector product by forming a matrix with two rows and lots of columns.

```
% Reshape into a matrix with 2 rows and  
% floor(length(x)/2) columns.  
n = 2*floor(length(x)/2);  
X = reshape(x(1:n),2,n/2);
```

All this preparation has been so that we can do the actual finite field arithmetic quickly and easily in Matlab.

```
% Encode with matrix multiplication modulo p.  
A = [71 2; 2 26];  
Y = mod(A*X,p);
```

Finally, convert the numbers back to printable characters.

```
% Reshape into a single row.  
y = reshape(Y,1,n);
```

```
% If length(x) is odd, encode the last character.  
% Recall that (p-1) is its own inverse!  
if length(x) > n  
    y(n+1) = mod((p-1)*x(n+1),p);  
end  
% Convert to printable ASCII characters.  
% Convert back to characters.  
y = char(y+32);  
y(y==127) = c1;  
y(y==128) = c2;
```

As an example, let's follow the computation of  $y = \text{crypto97}(\text{'Hello world'})$ . We begin with a character string.

```
x = 'Hello world'
```

This is converted to an integer vector.

```
x =  
    40  69  76  76  79  0  87  79  82  76  68
```

The `length(x)` is odd, so the reshaping temporarily ignores the last element.

```
X =  
    40 76 79 87 82  
    69 76  0 79 76
```

A conventional matrix-vector multiplication  $A * X$  produces an intermediate matrix.

```
    2978 5548 5609 6335 5974  
    1874 2128  158 2228 2140
```

Then the `mod(.,p)` operation produces

```
Y =  
    68 19 80 30 57  
    31 91 61 94  6
```

This is rearranged to a row vector.

```
y =  
    68 31 19 91 80 61 30 94 57 6
```

Now the last element of `x` is encoded by itself and attached to the end of `y`.

```
y =  
68 31 19 91 80 61 30 94 57 6 29
```

Finally,  $y$  is converted back to a character string to produce the encrypted result.

```
y = 'd?3{p]>~Y&='
```

If we now compute  $\text{crypto97}(y)$ , we get back our original 'Hello world' message.

---

## References

- [1] Most of the materials here are adopted from C. Moler, *Numerical Computing with Matlab* at the *Mathworks site*.
- [2] G. Birkhoff, G. and S. MacLane, *A Survey of Modern Algebra*, 5th edition New York: Macmillan, p. 413, 1996.
- [3] *D. Stinson Cryptography: Theory and Practice*, 2nd edition, CRC press, 2002.