

Cryptography

Team members

Shorouq Abd-elhameid Hussein	202001473	s-sherouk.abgelhamid@zewailcity.edu.eg
Fatma Taha	202000354	s-fatma.abdalgalil@zewailcity.edu.eg

Introduction

I) What is cryptography?

Cryptography is a method of protecting information and communications by encoding it in a way that only the people who need to know can interpret and process it. As a result, unwanted access to information is prevented.

The suffix "graphy" means "writing" and the prefix "crypt" means "hidden."

The procedures used to protect the data in cryptography are derived from mathematical principles and a set of rule-based calculations known as algorithms that change data in ways that make them difficult to decode. These algorithms are used to generate cryptographic keys, digitally sign documents, verify data privacy, browse the internet, and protect sensitive transactions such as credit card and debit card transactions.

Cryptography is often connected with the process of converting plain text to ciphertext, which is text encoded in such a way that only the intended receiver of the text can decode it, a process known as encryption, and a process of converting the ciphertext into plain text which is known as decryption.

II) Features Of Cryptography

1- Confidentiality:

Only the person, for whom the information is intended, has access to it, and no one else has access to this data.

2- Integrity:

Information cannot be modified in storage or transition between sender and intended receiver without any addition to information being detected.

3- Non-repudiation:

The creator/sender of information cannot deny his or her intention to send information at a later stage.

4- Authentication:

The identities of sender and receiver are confirmed. As well as destination/origin of the information is confirmed.

III) Types of cryptography

1- Symmetric Key Cryptography:

In this type, the sender and receiver of a message use a single common key to encrypt and decrypt messages.

2- Hash Functions:

This algorithm does not make use of any keys. A hash value with a fixed length is calculated based on the plain text, making it impossible to reconstruct the plain text's contents. Hash functions are used by several operating systems to encrypt passwords.

3- Asymmetric Key Cryptography:

Asymmetric cryptography, often known as public-key cryptography, encrypts and decrypts data using a pair of related keys (one public key and one private key) to prevent unwanted access or usage.

Hill cipher method

I) What is the Hill cipher method?

Hill cipher is a symmetrical encryption substitution technique developed by Lester Hill in 1929.

The advantage of this technique is that it can encrypt a group of letters like two or three letters at a time using one key.

The key here is a square and nonsingular matrix, if the matrix is $(n \times n)$ then (n) plaintext letters can be encrypted at the same time.

II) How Hill cipher technique works?

- To convert plain text into ciphertext (encryption process) using the Hill cipher technique the following steps must be followed:

- 1- Specify your key which is an $(n \times n)$ invertible matrix, and (n) rely on the number of plaintext letters that are intended to encrypt.
- 2- Replace each plaintext letter with its numerical value, as each letter has a corresponding numerical value according to (table 1).

A	B	C	D	E	F	G	H	I	J	K	L	M
O	1	2	3	4	5	6	7	8	9	10	11	12
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

(T.1)

- 3- Form $(n \times 1)$ column vector P since P is the column vector of plaintext numeric values.
- 4- Using equation one a ciphertext (C) could be created.

$$C = AP \bmod N \quad (\text{eq.1})$$

N: is the number of alphabets which is 26.

A: is $(n \times n)$ invertible matrix.

- To convert ciphertext into plaintext (decryption process) the following steps must be followed:
 - 1- Find the inverse of the matrix.
 - 2- Using the same table, we used during the encryption process, to replace each letter of the ciphertext with its numerical value.
 - 3- Using equation 2 get P and convert each plaintext vector to its letter using table 1.

$$P = A^{-1}C \bmod N \quad (\text{eq.2})$$

In general:

If $A = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix}$ and $P = \begin{pmatrix} p_{11} \\ \vdots \\ p_{n1} \end{pmatrix}$ then we get in the encryption process

$$\begin{pmatrix} c_{11} \\ \vdots \\ c_{n1} \end{pmatrix} = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} p_{11} \\ \vdots \\ p_{n1} \end{pmatrix} \bmod N.$$

The advantage of this method is that it will be difficult to decrypt the data when the size of the matrix is increased.

III) Using more than one key:

- As we mentioned above that matrix is the key used to decode or encode the data ,we could make use of some properties of matrices like the associative property as this will enable us to use more than one key to encipher and decipher the data which make the coding process more complicated.

- You must take care that the matrix you chose to be your key should be invertible or it would be impossible to decrypt your data.

Let A,B are two matrices as ciphers.

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix}, B = \begin{pmatrix} b_{11} & \cdots & b_{1n} \\ \vdots & \ddots & \vdots \\ b_{n1} & \cdots & b_{nn} \end{pmatrix}, P = \begin{pmatrix} p_{11} \\ \vdots \\ p_{n1} \end{pmatrix}$$

$$C \equiv ABP = A(BP) = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} b_{11} & \cdots & b_{1n} \\ \vdots & \ddots & \vdots \\ b_{n1} & \cdots & b_{nn} \end{pmatrix} \begin{pmatrix} p_{11} \\ \vdots \\ p_{n1} \end{pmatrix} = \begin{pmatrix} c_{11} \\ \vdots \\ c_{n1} \end{pmatrix} \mod N$$

And the decryption algorithm is just get the inverse of AB and apply equation 3

$$P \equiv (AB)^{-1}C \equiv B^{-1}A^{-1}C = B^{-1}(A^{-1}C) \mod N \quad (\text{eq.3})$$

so we could conclude that n number of invertible matrices could be used to encode or decode the data according to the following two equations:

For encryption:

$$C \equiv (ABC \dots M)P \mod N$$

For decryption:

$$P \equiv (ABC \dots M)^{-1}C \mod N$$

IV) Examples:

EX 1)

Consider the message 'ACT', and the key below:

$$\begin{pmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{pmatrix}$$

Since 'A' is 0, 'C' is 2 and 'T' is 19, the message is the vector:

$$\begin{pmatrix} 0 \\ 2 \\ 19 \end{pmatrix}$$

Thus the enciphered vector is given by:

$$\begin{pmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{pmatrix} \begin{pmatrix} 0 \\ 2 \\ 19 \end{pmatrix} = \begin{pmatrix} 67 \\ 222 \\ 319 \end{pmatrix} = \begin{pmatrix} 15 \\ 14 \\ 7 \end{pmatrix} \pmod{26}.$$

which corresponds to a ciphertext of 'POH'

Now, suppose that our message is instead 'CAT', or

$$\begin{pmatrix} 2 \\ 0 \\ 19 \end{pmatrix}$$

This time, the enciphered vector is given by

$$\begin{pmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{pmatrix} \begin{pmatrix} 2 \\ 0 \\ 19 \end{pmatrix} = \begin{pmatrix} 31 \\ 216 \\ 325 \end{pmatrix} = \begin{pmatrix} 5 \\ 8 \\ 13 \end{pmatrix} \pmod{26}$$

which corresponds to a ciphertext of 'FIN'.

EX 2)

$$\text{Let } K = \begin{pmatrix} 3 & 3 \\ 2 & 5 \end{pmatrix}$$

be the key and suppose the plaintext message is 'HELP'. Then this plaintext is represented by two pairs.

$$\text{HELP} \rightarrow \begin{pmatrix} H \\ E \end{pmatrix}, \begin{pmatrix} L \\ P \end{pmatrix} \rightarrow \begin{pmatrix} 7 \\ 4 \end{pmatrix}, \begin{pmatrix} 11 \\ 15 \end{pmatrix}.$$

Then we compute

$$\begin{pmatrix} 3 & 3 \\ 2 & 5 \end{pmatrix} \begin{pmatrix} 7 \\ 4 \end{pmatrix} = \begin{pmatrix} 7 \\ 8 \end{pmatrix} \pmod{26} \text{ and,}$$

$$\begin{pmatrix} 3 & 3 \\ 2 & 5 \end{pmatrix} \begin{pmatrix} 11 \\ 15 \end{pmatrix} = \begin{pmatrix} 0 \\ 19 \end{pmatrix} \pmod{26}$$

and continue encryption as follows:

$$\begin{pmatrix} 7 \\ 8 \end{pmatrix}, \begin{pmatrix} 0 \\ 19 \end{pmatrix} \rightarrow \begin{pmatrix} H \\ I \end{pmatrix}, \begin{pmatrix} A \\ T \end{pmatrix}$$

To decrypt, get the inverse of K

$$(K^{-1} = \begin{pmatrix} 15 & 17 \\ 20 & 9 \end{pmatrix} \pmod{26})$$

$$\text{HIAT} \rightarrow \begin{pmatrix} H \\ I \end{pmatrix}, \begin{pmatrix} A \\ T \end{pmatrix} \rightarrow \begin{pmatrix} 7 \\ 8 \end{pmatrix}, \begin{pmatrix} 0 \\ 19 \end{pmatrix}$$

Then we compute

$$\begin{pmatrix} 15 & 17 \\ 20 & 9 \end{pmatrix} \begin{pmatrix} 7 \\ 8 \end{pmatrix} = \begin{pmatrix} 241 \\ 212 \end{pmatrix} = \begin{pmatrix} 7 \\ 4 \end{pmatrix} \pmod{26}$$

$$\begin{pmatrix} 15 & 17 \\ 20 & 9 \end{pmatrix} \begin{pmatrix} 0 \\ 19 \end{pmatrix} = \begin{pmatrix} 323 \\ 171 \end{pmatrix} = \begin{pmatrix} 11 \\ 15 \end{pmatrix} \pmod{26}.$$

Therefore,

$$\begin{pmatrix} 7 \\ 4 \end{pmatrix}, \begin{pmatrix} 11 \\ 15 \end{pmatrix} \rightarrow \begin{pmatrix} H \\ E \end{pmatrix}, \begin{pmatrix} L \\ P \end{pmatrix}.$$

Appendix:

Software: (python using NumPy)

Representing every letter by a unique number

```
Letter = ["A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K", "L", "M", "N", "O",  
         "P", "Q", "R", "S", "T", "U", "V", "W", "X", "Y", "Z"]  
Number = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,  
         15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25]
```

function to encrypt and decrypt a word of three letters By an argument x encrypting when it's equal 1 and decrypting when it's equal 2

```
def encryption(a, x):  
    alist = []  
    newlist = []  
    nlist = []  
    # for turning the input to list  
    for i in a:  
        alist.append(i)  
    for j in range(len(alist)):  
        for i in range(len(Letter)):  
            # print(alist[j], Letter[i])  
            if alist[j] == Letter[i]:  
                newlist.append(Number[i])  
  
    if x == 1:  
        key = [[6, 24, 1], [13, 16, 10], [20, 17, 15]]  
    else:  
        key = [[8, 5, 10], [21, 8, 21], [21, 12, 8]]  
  
    finalarray = np.dot(key, newlist)  
    for i in range(len(finalarray)):  
        while finalarray[i] >= 26:  
            finalarray[i] -= 26
```

```

    for j in range(len(finalarray)):
        for i in range(len(Number)):
            # print(alist[j], Letter[i])
            if finalarray[j] == Number[i]:
                nlist.append(Letter[i])

    return "".join(nlist)

```

Function to encrypt and decrypt a word of four letters by an argument x encrypting when it's equal 1 and decrypting when it's equal 2

```

def encryptionfourword(a, x):
    alist1 = []
    alist2 = []
    alist1n = []
    alist2n = []
    nlist1 = []
    nlist2 = []

    for i in range(4):
        if i > 1:
            alist2.append(a[i])
        else:
            alist1.append(a[i])

    # print(alist1)
    # print(alist2)

    for j in range(len(alist1)):
        for i in range(len(Letter)):
            # print(alist[j], Letter[i])
            if alist1[j] == Letter[i]:
                alist1n.append(Number[i])

    for j in range(len(alist2)):
        for i in range(len(Letter)):
            # print(alist[j], Letter[i])

```

```

        if alist2[j] == Letter[i]:
            alist2n.append(Number[i])
# print(alist1n)
# print(alist2n)
if x == 1:
    key = [[3, 3], [2, 5]]
else:
    key = [[15, 17], [20, 9]]

finalarray1 = np.dot(key, alist1n)
for i in range(len(finalarray1)):
    while finalarray1[i] >= 26:
        finalarray1[i] -= 26
for j in range(len(finalarray1)):
    for i in range(len(Number)):
        # print(alist[j], Letter[i])
        if finalarray1[j] == Number[i]:
            nlist1.append(Letter[i])
finalarray2 = np.dot(key, alist2n)
for i in range(len(finalarray2)):
    while finalarray2[i] >= 26:
        finalarray2[i] -= 26
for j in range(len(finalarray2)):

```

```

        while finalarray2[i] >= 26:
            finalarray2[i] -= 26
    for j in range(len(finalarray2)):
        for i in range(len(Number)):
            # print(alist[j], Letter[i])
            if finalarray2[j] == Number[i]:
                nlist2.append(Letter[i])
nlist1 = "".join(nlist1)
nlist2 = "".join(nlist2)

finalarray = [nlist1, nlist2]
return "".join(finalarray)

```

The main script code to call the functions above by the number of letters the user enters

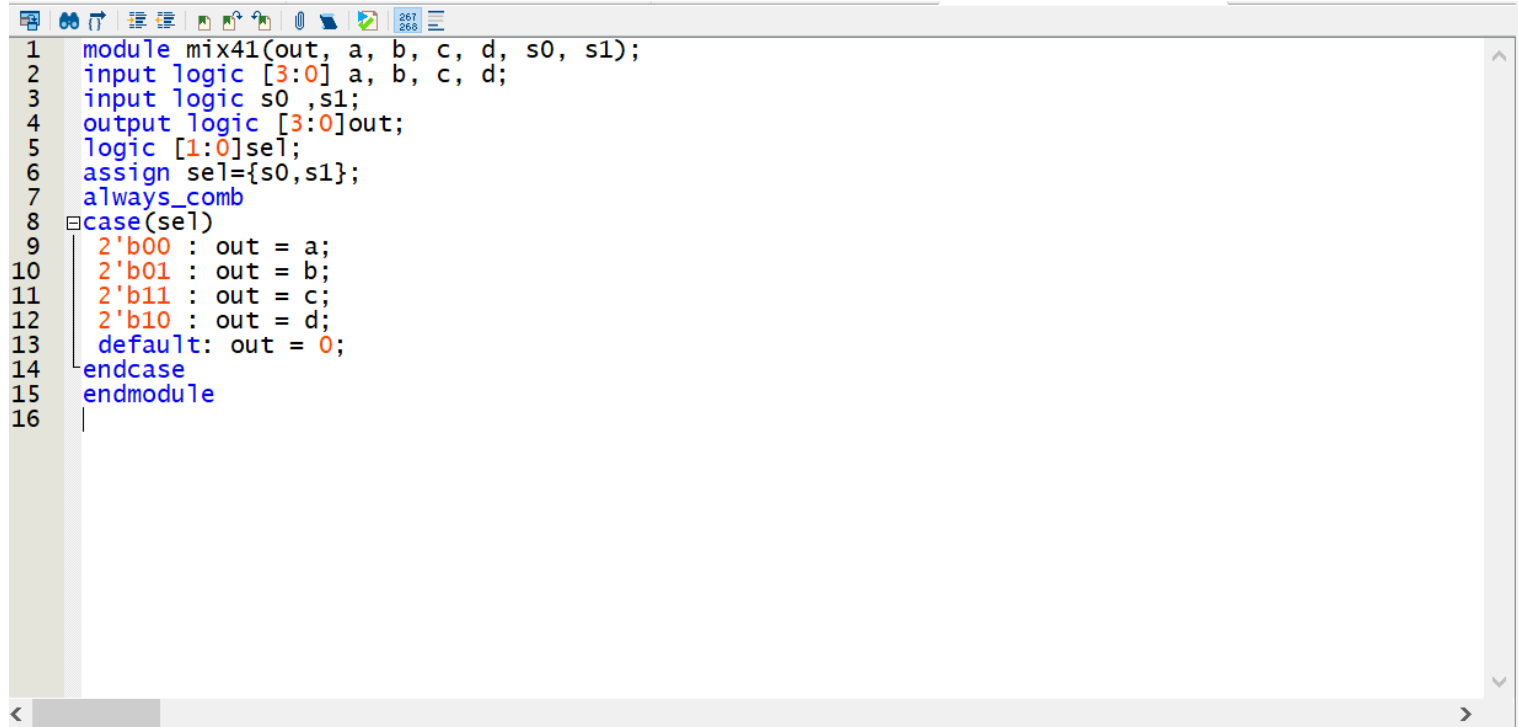
```
boo = "YES"
while boo == "YES":
    a = input("enter your word")
    c = 0
    for i in a:
        c += 1

    ask = int(input("enter 1 if you want to encrypt and 2 if you want to decrypt"))
    if c == 3:
        if ask == 1:
            print(encryption(a, 1))
        else:
            print(encryption(a, 2))
    elif c == 4:
        if ask == 1:
            print(encryptionfourword(a, 1))
        else:
            print(encryptionfourword(a, 2))

    boo = input("do you want to try again")
```

Hardware:

4 to 1 mux module to switch between the different inputs depending on the Selectors t



```
1 module mix41(out, a, b, c, d, s0, s1);
2   input logic [3:0] a, b, c, d;
3   input logic s0, s1;
4   output logic [3:0] out;
5   logic [1:0] sel;
6   assign sel={s0,s1};
7   always_comb
8   case(sel)
9     2'b00 : out = a;
10    2'b01 : out = b;
11    2'b11 : out = c;
12    2'b10 : out = d;
13    default: out = 0;
14  endcase
15 endmodule
16
```

Display-hex-mux model to switch between the different seven segments and display the letters by hex encoding

```

module disp_hex_mux
(
input logic clk, reset,
input logic [3:0] hex3, hex2, hex1, hex0, // hex digits
input logic [3:0] dp_in, // 4 decimal points
output logic [3:0] an, // enable 1- out- of- 4 asserted low
output logic [7:0] sseg // led segments
);
// constant declaration
// refreshing rate around 800 Hz (50 MHz/2^16)
localparam N = 18; // real
//localparam N = 4; //simulation
// internal signal declaration
logic [N-1:0] q_reg;
wire [N-1:0] q_next;
logic [3:0] hex_in;
logic dp;
// N- bit counter
// register
always_ff @(posedge clk, posedge reset)
if (reset)
q_reg <= 0;
else
q_reg <= q_next;
// next - state logic
assign q_next = q_reg + 1;
// 2 MSBs of counter to control 4- to- 1 multiplexing
// and to generate active- low enable signal
always_comb
case (q_reg[N-1:N-2])

```

```

always_comb
case (q_reg[N-1:N-2])
2'b00:
begin
an = 4'b0001; //e hex
hex_in = hex0;
dp = dp_in[0];
end
2'b01:
begin
an = 4'b0010; //d hex
hex_in = hex1;
dp = dp_in[1];
end
2'b10:
begin
an = 4'b0100; //b hex
hex_in = hex2;
dp = dp_in[2];
end
default :
begin
an = 4'b1000; //7 hex
hex_in = hex3;
dp = dp_in[3];
end
endcase
// hex to seven -segment led display
always_comb

```

```

    an = 4'b1000; //7 hex
    hex_in = hex3;
    dp = dp_in[3];
end
endcase
// hex to seven -segment led display
always_comb
begin
case(hex_in) // abcdefg
4'h0: sseg[6:0] = 7'b0001000; // 01 in hex A
4'h1: sseg[6:0] = 7'b0010000; // 4F in hex
4'h2: sseg[6:0] = 7'b0110000; // 12 in hex E
4'h3: sseg[6:0] = 7'b1001000; // 06 in hex H
4'h4: sseg[6:0] = 7'b1110001; // 4C in hex L
4'h5: sseg[6:0] = 7'b0011000; // 24 in hex P
4'h6: sseg[6:0] = 7'b1111001; // 20 in hex i
4'h7: sseg[6:0] = 7'b1110000; // 0F in hex t
4'h8: sseg[6:0] = 7'b0001001; // 00 in hex N
4'h9: sseg[6:0] = 7'b0000001; // 04 in hex f
4'ha: sseg[6:0] = 7'b0001000; // 08 in hex R
4'hb: sseg[6:0] = 7'b1100000; // 60 in hex
4'hc: sseg[6:0] = 7'b0110001; // 31 in hex
4'hd: sseg[6:0] = 7'b1000010; // 42 in hex
4'he: sseg[6:0] = 7'b0110000; // 30 in hex
default: sseg[6:0] = 7'b0111000; //4â€™hf character 38 in hex
endcase
end
endmodule

```

The top level model to control witch word will be executed and displayed in the seven segments

```

module first (reset,clk,s,an,sseg);
input logic reset ,clk;
input logic [1:0]s;
output logic [3:0]an;
output logic [7:0]sseg;
logic [3:0]first,second,third,forth;
not (resetbar,reset);
mix41 firstletter(first,4'b0011,4'b0011,4'b0101,4'b0000,s[0],s[1]); //mix41(out, a, b, c, d, s[0], s[1]);
mix41 secondletter(second,4'b0010,4'b0110,4'b0100,4'b0011,s[0],s[1]);
mix41 thiredletter(third,4'b0100,4'b0000,4'b0000,4'b1000,s[0],s[1]);
mix41 forthletter(forth,4'b0101,4'b0111,4'b1000,4'b1000,s[0],s[1]);
disp_hex_mux out1 (clk,resetbar,first,second,third,forth,4'b1111,an,sseg);
endmodule

```


Conclusion

Now we could conclude the importance of linear algebra in safeguarding data and protect people's privacy as it contributes with a big rule and thanks to Hill and its efforts we now be able to using our credit cards safely and send messages to each others keeping our privacy safe.

References

- (What is the Hill cipher?, 2022).
- Salomaa, Arto. Public-key cryptography. Turku, Finland. 2nd edition.
- Lester S. Hill, Concerning Certain Linear Transformation Apparatus of Cryptography, *The American Mathematical Monthly* Vol.38, 1931, pp. 135–154.

