CSCI361 Cryptography – Assignment 1
Name : LEE CHIA LIN
Student ID : 6097881

Part 1. Affine Cipher

1. C = aM + b (mod 26)
   b can be any number in the range of 0 to 25.
   a has to be relatively prime to the modular, in this case, relatively prime to 26.
   This is because M will not be able to be uniquely decoded if a has a common factor with 26.
   For example, try key (13, 1) with two different M, M = 1, M = 3.
   13(1) + 1 mod 26 = 14 mod 26
   13(3) + 1 mod 26 = 14 mod 26

   Therefore,
   (i) (3, 9) is valid.
   3 is relatively prime to 26, and 9 is in the range of 0 to 25.

   (ii) (6, 4) is invalid.
   6 is not relatively prime to 26. A common factor is 2.

   (iii) (11, 0) is valid.
   11 is relatively prime to 26, and 0 is in the range of 0 to 25.

   (iv) (0, 13) is invalid.
   0 is not relatively prime to 26. A common factor is 0.

   (v) (13, 1) is invalid.
   13 is not relatively prime to 26. A common factor is 13.

2. $M = (C - b)a^{-1} mod\ 26$

   (i) (3, 9)
   $M = (C - 9)3^{-1} mod\ 26$
   Use extended euclidean algorithm to find $3^{-1} mod\ 26$…

n1	26

n2	3

n1	**n2**	**r**	**q**	**c1**	**d1**	**c2**	**d2**
26	3	2	8	1	0	0	1
3	2	1	1	0	1	1	-8
2	**1**	0	2	1	-8	**-1**	**9**

   Therefore, from the working table above,
   gcd(26, 3) = 1
   1 = 26*(-1) + 3*(9)
   Decryption function formula: $\boldsymbol{M = (C - 9)9\ mod\ 26}$

(ii) (11, 0)

$M = (C - 0)11^{-1} mod\ 26$

Use extended euclidean algorithm to find $11^{-1} mod\ 26$...

n1	26

n2	11

n1	**n2**	**r**	**q**	**c1**	**d1**	**c2**	**d2**
26	11	4	2	1	0	0	1
11	4	3	2	0	1	1	-2
4	3	1	1	1	-2	-2	5
3	**1**	0	3	-2	5	**3**	**-7**

Therefore, from the working table above,

gcd(26, 11) = 1

1 = 26*(3) + 11*(-7)

-7+26 = 19

Decryption function formula: $\boldsymbol{M = (C - 0)19\ mod\ 26}$

3. javac Part1.java
   java Part1

   Runtime SAMPLE

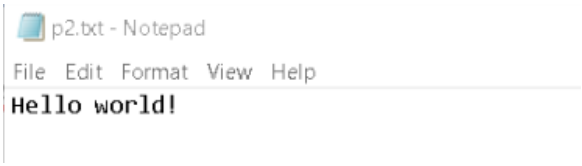
   Y

   13 1 encrypt p2.txt p3.txt

```
C:\Users\Chia Lin\Desktop\LEECHIALIN_6097881_CSCI361_A1>java Part1
Do you have the key? Y/N:
Y

Please enter key, encrypt or decrypt, input file name and output file name:
SAMPLE: 3 8 encrypt fileIn.txt fileOut.txt
13 1 encrypt p2.txt p3.txt
Invalid key!
```

4. Unencrypted textfile p2.txt



<u>Screenshot result – (3, 9)</u>

Encrypt

```
Do you have the key? Y/N:
Y

Please enter key, encrypt or decrypt, input file name and output file name:
SAMPLE: 3 8 encrypt fileIn.txt fileOut.txt
3 9 encrypt p2.txt p3.txt
Successfully saved to p3.txt
```

Encrypted textfile

p3.txt - Notepad

File Edit Format View Help

Wniir praik!

Decrypt with known keys

```
Do you have the key? Y/N:
Y

Please enter key, encrypt or decrypt, input file name and output file name:
SAMPLE: 3 8 encrypt fileIn.txt fileOut.txt
3 9 decrypt p3.txt p4.txt
Successfully saved to p4.txt
```

<u>Screenshot result – (11, 0)</u>

Encrypt

```
Do you have the key? Y/N:
Y

Please enter key, encrypt or decrypt, input file name and output file name:
SAMPLE: 3 8 encrypt fileIn.txt fileOut.txt
11 0 encrypt p2.txt p3.txt
Successfully saved to p3.txt
```

Encrypted textfile

p3.txt - Notepad

File Edit Format View Help

Zsrry iyfrh!

Decrypt with known keys

```
Do you have the key? Y/N:
Y

Please enter key, encrypt or decrypt, input file name and output file name:
SAMPLE: 3 8 encrypt fileIn.txt fileOut.txt
11 0 decrypt p3.txt p4.txt
Successfully saved to p4.txt
```

Part 2. Encryption Question

Using the program from Part1… **Decrypting without known key (brute force)**

```
C:\Users\Chia Lin\Desktop\LEECHIALIN_6097881_CSCI361_A1>java Part1
Do you have the key? Y/N:
N

Please input file name to be decrypted and output file name:
SAMPLE: fileIn.txt fileOut.txt
Assignment-Part2.txt myDecrypt.txt
Successfully decrypted with keys (5, 4)
```

Decrypted textfile myDecrypt.txt

myDecrypt.txt - Notepad

File Edit Format View Help

```
Implement the index of coincidence and the mutual index of coincidence to compute the following:
    - the index of coincidence of text A.
    - the mutual index of coincidence of texts A and B.

Remember to submit all programming codes that you wrote for doing this task.

You will not be given marks if you just give those indices without proper codes.

Text A: cyber security is about how we develop secure computers and computer networks, to ensure that the data stored and tı

Text B: cyber security has been identified as one of the strategic priorities in australia to meet the demands of law enforc
```

Finding the IC of text A, and MIC of text A and B.

javac Part2.java

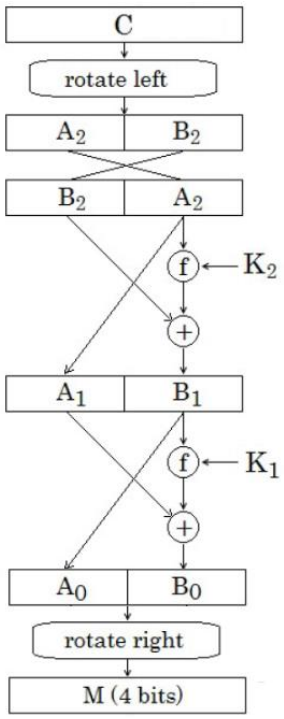
I saved text A into part2-a.txt and text B into part2-b.txt

```
C:\Users\Chia Lin\Desktop\LEECHIALIN_6097881_CSCI361_A1>java Part2
Please enter filename of textA and textB:
part2-a.txt part2-b.txt

Successfully read from part2-a.txt
Successfully read from part2-b.txt
------------------------------------------------------------
Index of Coincidence (IC) of part2-a.txt: 0.06693409742120345
Index of Coincidence (IC) of part2-b.txt: 0.07041499330655958
Mutual IC of textA and textB: 0.06930120481927711
```

Part 3. LDES

1.



2. and 3.

javac Part3.java

```
C:\Users\Chia Lin\Desktop\LEECHIALIN_6097881_CSCI361_A1>java Part3
Encryption table
   Key  0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
   00   1001 1000 0011 0010 1100 1101 0110 0111 0001 0000 1011 1010 0100 0101 1110 1111
   01   1111 1110 0101 0100 1010 1011 0000 0001 0111 0110 1101 1100 0010 0011 1000 1001
   10   0110 0111 1100 1101 0011 0010 1001 1000 1110 1111 0100 0101 1011 1010 0001 0000
   11   0000 0001 1010 1011 0101 0100 1111 1110 1000 1001 0010 0011 1101 1100 0111 0110
-------------------------------------------------------------------------------------------
Decryption table
   Key  0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
   00   1001 1000 0011 0010 1100 1101 0110 0111 0001 0000 1011 1010 0100 0101 1110 1111
   01   0110 0111 1100 1101 0011 0010 1001 1000 1110 1111 0100 0101 1011 1010 0001 0000
   10   1111 1110 0101 0100 1010 1011 0000 0001 0111 0110 1101 1100 0010 0011 1000 1001
   11   0000 0001 1010 1011 0101 0100 1111 1110 1000 1001 0010 0011 1101 1100 0111 0110
-------------------------------------------------------------------------------------------
Verifying E(1100) = E(1000) + E(0100) + E(0000) with the keys 00, 01, 10, 11
E(1000) + E(0100) + E(0000) = 0100,     E(1100): 0100   Verified
E(1000) + E(0100) + E(0000) = 0010,     E(1100): 0010   Verified
E(1000) + E(0100) + E(0000) = 1011,     E(1100): 1011   Verified
E(1000) + E(0100) + E(0000) = 1101,     E(1100): 1101   Verified
-------------------------------------------------------------------------------------------
Form equation and check with keys for E(1010)
E(1000) + E(0010) + E(0000) = 1011,     E(1010) = 1011  Verified
E(1000) + E(0010) + E(0000) = 1101,     E(1010) = 1101  Verified
E(1000) + E(0010) + E(0000) = 0100,     E(1010) = 0100  Verified
E(1000) + E(0010) + E(0000) = 0010,     E(1010) = 0010  Verified
-------------------------------------------------------------------------------------------
Form equation and check with keys for E(1001)
E(1000) + E(0001) + E(0000) = 0000,     E(1001) = 0000  Verified
E(1000) + E(0001) + E(0000) = 0110,     E(1001) = 0110  Verified
E(1000) + E(0001) + E(0000) = 1111,     E(1001) = 1111  Verified
E(1000) + E(0001) + E(0000) = 1001,     E(1001) = 1001  Verified
-------------------------------------------------------------------------------------------
Form equation and check with keys for E(0110)
E(0100) + E(0010) + E(0000) = 0110,     E(0110) = 0110  Verified
E(0100) + E(0010) + E(0000) = 0000,     E(0110) = 0000  Verified
```

Part 4. MDES

4. and 5.

javac Part4_sbox.java

```
C:\Users\Chia Lin\Desktop\LEECHIALIN_6097881_CSCI361_A1>java Part4_sbox
After changing the SBOX...
-----------------------------------------------------------------------------------------
Encryption table
   Key  0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
   00   0000 0100 1000 1100 0001 0101 1111 1011 0010 1001 1010 0111 0011 1110 1101 0110
   01   0110 0010 1000 1100 0001 0101 1001 1101 0100 1111 1010 0111 0011 1110 1011 0000
   10   1111 0100 0001 1100 1000 0101 0000 1011 0010 0110 1010 1110 0011 0111 1101 1001
   11   1001 0010 0001 1100 1000 0101 0110 1101 0100 0000 1010 1110 0011 0111 1011 1111
-----------------------------------------------------------------------------------------
Decryption table
   Key  0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
   00   0000 0100 1000 1100 0001 0101 1111 1011 0010 1001 1010 0111 0011 1110 1101 0110
   01   1111 0100 0001 1100 1000 0101 0000 1011 0010 0110 1010 1110 0011 0111 1101 1001
   10   0110 0010 1000 1100 0001 0101 1001 1101 0100 1111 1010 0111 0011 1110 1011 0000
   11   1001 0010 0001 1100 1000 0101 0110 1101 0100 0000 1010 1110 0011 0111 1011 1111
-----------------------------------------------------------------------------------------
Verifying that question 3 equations are no longer valid...
-----------------------------------------------------------------------------------------
Verifying E(1100) = E(1000) + E(0100) + E(0000) with the keys 00, 01, 10, 11
E(1000) + E(0100) + E(0000) = 0011,     E(1100): 0011   Verified
E(1000) + E(0100) + E(0000) = 0011,     E(1100): 0011   Verified
E(1000) + E(0100) + E(0000) = 0101,     E(1100): 0011   Not verified
E(1000) + E(0100) + E(0000) = 0101,     E(1100): 0011   Not verified
-----------------------------------------------------------------------------------------
Form equation and check with keys for E(1010)
E(1000) + E(0010) + E(0000) = 1010,     E(1010) = 1010  Verified
E(1000) + E(0010) + E(0000) = 1010,     E(1010) = 1010  Verified
E(1000) + E(0010) + E(0000) = 1100,     E(1010) = 1010  Not verified
E(1000) + E(0010) + E(0000) = 1100,     E(1010) = 1010  Not verified
-----------------------------------------------------------------------------------------
Form equation and check with keys for E(1001)
E(1000) + E(0001) + E(0000) = 0110,     E(1001) = 1001  Not verified
```

6.

javac Part4_ecb_cbc.java

Sample input

ECB

10 encrypt f4a5a32

ECB encrypt

```
C:\Users\Chia Lin\Desktop\LEECHIALIN_6097881_CSCI361_A1>java Part4_ecb_cbc
*************All input are CASE sensitive**********
What are you entering? (ECB/CBC): ECB

Enter key, encrypt/decrypt, hex string:
SAMPLE: 01 decrypt b6f7a11
10 encrypt f4a5a32

Message: f4a5a32
Ciphertext: 98a5ac1
```

ECB decrypt

```
C:\Users\Chia Lin\Desktop\LEECHIALIN_6097881_CSCI361_A1>java Part4_ecb_cbc
*************All input are CASE sensitive**********
What are you entering? (ECB/CBC): ECB

Enter key, encrypt/decrypt, hex string:
SAMPLE: 01 decrypt b6f7a11
01 decrypt b6f7a11

Message: b6f7a11
Ciphertext: e09ba44
```

CBC encrypt

```
*************All input are CASE sensitive**********
What are you entering? (ECB/CBC): CBC

Enter key, iv(0-f), encrypt/decrypt, hex string:
SAMPLE: 11 a encrypt 2a45def
11 a encrypt 2a45def

Message: 2a45def
Ciphertext: 4bfadcc
Press any key to continue...
```

CBC decrypt

```
*************All input are CASE sensitive**********
What are you entering? (ECB/CBC): CBC

Enter key, iv(0-f), encrypt/decrypt, hex string:
SAMPLE: 11 a encrypt 2a45def
00 4 decrypt b412ab2

Message: b412ab2
Ciphertext: 3a098d3
Press any key to continue...
```

Part 5. OFB

Section 1 and 2

javac Part5.java

```
C:\Users\Chia Lin\Desktop\LEECHIALIN_6097881_CSCI361_A1>java Part5
Section 2 Encrypt student number using 4-bit
--------------------------------------------------------------------------------
4-bit encrypt 6097881
Round
 0      cipher: 0000000000000000000000000000000000000000000000000000000000001010
 1      cipher: 0000000000000000000000000000000000000000000000000000000010101100
 2      cipher: 0000000000000000000000000000000000000000000000000000000101011000000
 3      cipher: 0000000000000000000000000000000000000000000000000010101100000000000
 4      cipher: 0000000000000000000000000000000000000000000000010101100000000001100
 5      cipher: 0000000000000000000000000000000000000000000010101100000000000011001110
 6      cipher: 0000000000000000000000000000000000000000101011000000000000110011100011
 7      cipher: 0000000000000000000000000000000000001010110000000000001100111000111110
 8      cipher: 0000000000000000000000000000000010101100000000000110011100011111011111
 9      cipher: 0000000000000000000000000000101011000000000001100111000111110111111101
 10     cipher: 0000000000000000000000101011000000000001100111000111110111111010010
 11     cipher: 0000000000000000000101011000000000001100111000111110111111010010101000
 12     cipher: 0000000000000000101011000000000001100111000111110111111010010100001001
 13     cipher: 0000000000000101011000000000001100111000111110111111010010100010010101
 14     cipher: 0000000000101011000000000001100111000111110111111010010100010010101011010
 15     cipher: 1010110000000000110011100011111011111101001010001001010110100101

Cipher (Binary): 1010110000000000110011100011111011111101001010001001010110100101
Cipher: -6052611129255357019

4-bit (Binary): 1010
4-bit: 10
--------------------------------------------------------------------------------
```

Section 3

Same program as section 1 and 2, **Part5.java**

Output is displayed after Section 2's output

```
3-bit encrypt 6097881
Round
 0      cipher: 0000000000000000000000000000000000000000000000000000000000000101
 1      cipher: 0000000000000000000000000000000000000000000000000000000000101100
 2      cipher: 0000000000000000000000000000000000000000000000000000000101100011
 3      cipher: 0000000000000000000000000000000000000000000000000000101100011100
 4      cipher: 0000000000000000000000000000000000000000000000000101100011100001
 5      cipher: 0000000000000000000000000000000000000000000000101100011100001000
 6      cipher: 0000000000000000000000000000000000000000000101100011100001000101
 7      cipher: 0000000000000000000000000000000000000000101100011100001000101110
 8      cipher: 0000000000000000000000000000000000000101100011100001000101110011
 9      cipher: 0000000000000000000000000000000000101100011100001000101110011001
 10     cipher: 0000000000000000000000000000000101100011100001000101110011001011
 11     cipher: 0000000000000000000000000000101100011100001000101110011001011000
 12     cipher: 0000000000000000000000000101100011100001000101110011001011000000
 13     cipher: 0000000000000000000000101100011100001000101110011001011000000100
 14     cipher: 0000000000000000000101100011100001000101110011001011000000100100
 15     cipher: 0000000000000000101100011100001000101110011001011000000100100010
 16     cipher: 0000000000000101100011100001000101110011001011000000100100010101
 17     cipher: 0000000000101100011100001000101110011001011000000100100010101011
 18     cipher: 0000000101100011100001000101110011001011000000100100010101011011
 19     cipher: 0000101100011100001000101110011001011000000100100010101011011100
 20     cipher: 0101100011100001000101110011001011000000100100010101011011100110

Cipher (Binary): 1011000111000100010111001100101100000010010001010101101110011 0
Cipher: 6404425651844110054

3-bit (Binary): 010
3-bit: 2
Press any key to continue...
```

The number of rounds to loop was determined by 64 divided by number of bits.

Encrypting with c-bit will take a slightly longer time, as we can see the loop goes through more rounds to achieve because c-bit, in this case 3-bit, is smaller than 4-bit.

The loop only gets 3-bit per loop and takes longer to fill up the 64-bits.

Part 6. Synchronous Stream Cipher

a.

$c = m + k \ (mod\ 26)$

$(c - k) = m\ mod\ 26$

$(c - k) mod\ 26 = m$

The decryption algorithm remains the same as the encryption, except that instead of adding k and the input, we subtract k from the input.

b. and c. and d.

javac Part6.java

No user input required, text and key is hardcoded in program

```
C:\Users\Chia Lin\Desktop\LEECHIALIN_6097881_CSCI361_A1>java Part6
Part c, encrypt WOLLONGONG with key = 3
GMSMUDJFZE
-----------------------------------------------
Part d, decrypt MQJJ with key = 3
CSCI
-----------------------------------------------
```