Data Security and Cryptography Question

1. a) Decrypt using Casear Cipher. By default shift 3 letter, shift to right to encrypt and shift to left to decrypt.

- `Ciphertext:` `Zhofrph wr fubswrjudskb zruog`
- `Plaintext:` `Welcome to cryptography world`

b) Interpret secret message using Vigenere Cipher.

- `Ciphertext:` `RWXO XCPOJ`
- `Key:` `LOCKL LOCKL`
- `Plaintext:` `GIVE MONEY`

2. a) The input to S-box is 110111

Row – 11 » 3	Column – 1011 » 11	Output = 12

b) The input to S-box is 101010

Row – 10 » 2	Column – 0101 » 5	Output = 4

c) The input to S-box is 111101

Row – 11 » 3	Column – 1110 » 14	Output = 14

3. Explain if modification to the substitution cipher does provide added security?

This modification to the substitution cipher does not provide added security. This is because uppercase letters in English word rarely appear, such that it appears only at the beginning of words in a sentence. Substitution cipher can be easily break by using *frequency analysis* against all letter and focus on solving for the characters with the highest frequencies which still be the same lowercase letters. Once solved, there will be enough plaintext recovered to figure out all the uppercase letter in the ciphertext message.

4. How Alice can protect her email based on authenticity and integrity using digital signature?

Alice can protect her email based on authenticity and integrity using digital signature to digitally sign an email before sending to Bob. Bob can be certain that the message is really from Alice and it were not modified or tampered with from the time Alice signed it to the time Bob verified it. To achieve this, digital signature detect whether a message has been altered since it was completed and the former, to determine whether it was actually sent by the person or entity claimed to be the sender. Because the content is encrypted, any changes in the message will result in failure of the decryption with the appropriate key. Alice need to use **PGP** so that PGP can computes a hash *(message digest)* from the plaintext and then creates the digital signature from that hash using the sender's private key.

5. Given |P| = (-|M| - 128) mod 1024. Calculate the padding for SHA-512. Message = 8060 bits and 10500 bits.

```
|P| = (-|M| - 128) mod 1024
    = - 8060 - 128 mod 1024
    = - 8188 mod 1024
    =       _____8__
       1024 | - 8188
    = - | (- 8192)
    = - 8188 - (- 8192)
  P = 4
```

```
|P| = (-|M| - 128) mod 1024
    = - 10500 - 128 mod 1024
    = - 10628 mod 1024
    =       _____11__
       1024 | - 10628
    = - | -(- 10628)
    = - 10628 - (- 11264)
  P = 636
```

6. p = 11, q = 3, e = 3, calculate *n* and *ø(n)* and *d*.

```
n   = pq
    = 11 x 3
    = 33
ø(n) = (p-1) (q-1)
     = (11-1) (3-1)
     = (10) (2)
     = 20
```

e, 1 < e < ø(n), such that gcd (e, ø(n)) = 1

1 < 3 < 20   gcd (3, 20) = 1

Proof, gcd (3, 20) = 1

20 = 3.6 + 2

3 = 2.1 + 1

3 = 1.3 + 0

d, 1 < d < ø(n), such that ed = 1 mod ø(n)

3d = 1 mod 20

3d - 1 = 20

3d = 20 + 1

d = 21 / 3

Public key (n, e), Private key (d, p, q)

Public key (20, 3), Private key (7, 11, 3)

7. a) GCD (414, 662)

662 = 414 . 1 + 248

414 = 248 . 1 + 166

248 = 166 . 1 + 82

166 = 82 . 2 + 2

82 = 2 . 41 + 0

GCD (414, 662) = 2

b) GCD (939, 712)

939 = 712 . 1 + 227

712 = 227 . 3 + 31

227 = 31 . 7 + 10

31 = 10. 3 + 1

```
10  = 1 . 10 + 0
GCD (939, 712) = 1
```

8. Analyze the transformation of the new block cipher based on the initial add round key, the substitution byte, and the shift row.

- `Key:        59 65 73 20 79 6F 75 20 63 61 6E 20 64 6F 69 74`
- `Plaintext:  54 77 6F 20 4F 6E 65 20 4E 69 6E 65 20 54 77 6F`

State matrix (key)

w[0]	w[1]	w[2]	w[3]
59	79	63	64
65	6F	61	6F
73	75	6E	69
20	20	20	74

State matrix (plaintext)

w[0]	w[1]	w[2]	w[3]
54	4F	4E	20
77	6E	69	54
6F	65	6E	77
20	20	65	6F

Circular byte left shift w[3] = (64 6F 69 74) » (6F 69 74 64)

Substitution byte w[3] = (6F 69 74 64) » (96 C7 AC 7D)

- Add round key constant (XOR operation)
- (96 C7 AC 7D) XOR (01 00 00 00)

Sub byte	1001 0110	1100 0111	1010 1100
Round 01	0000 0001	0000 0000	0000 0000
XOR	1001 0111	1100 0111	1010 1100
g(w[3])	97	C7	AC

w[4] = g(w[3]) XOR w[0]

g(w[3])	1001 0111	1100 0111	1010 1100
w[0]	0101 1001	0110 0101	0111 0011
XOR	1100 1110	1010 0010	1101 1111
w[4]	CE	A2	DF

w[5] = w[4] XOR w[1]

w[4]	1100 1110	1010 0010	1101 1111
w[1]	0111 1001	0110 1111	0111 0101
XOR	1011 0111	1100 1101	1010 1010
w[5]	B7	CD	AA

w[6] = w[5] XOR w[2]

w[5]	1011 0111	1100 1101	1010 1010
W[2]	0110 0011	0110 0001	0110 1110
XOR	1101 0100	1010 1100	1100 0100
w[6]	D4	AC	C4

w[7] = w[6] XOR w[3]

w[6]	1101 0100	1010 1100	1100 0100
w[3]	0110 0100	0110 1111	0110 1001
XOR	1011 0000	1100 0011	1010 1101
w[7]	B0	C3	AD

- Combine w[4], w[5], w[6], w[7] to get 1st round key
- 1st round key = **(CE, A2, DF, 5D, B7, CD, AA, 7D, D4, AC, C4, 5D, B0, C3, AD, 29)**

State matrix

w[4]	w[5]	w[6]	w[7]
CE	B7	D4	B0
A2	CD	AC	C3
DF	AA	C4	AD
5D	7D	5D	29

Circular byte left shift w[7] = (B0 C3 AD 29) » (C3 AD 29 B0)

Substitution byte w[7] = (C3 AD 29 B0) » (10 AB 9B 44)

- Add round key constant (XOR operation)
- (10 AB 9B 44) XOR (02 00 00 00)

Sub byte	0001 0000	1010 1011	1001 1011
Round 02	0010 0000	0000 0000	0000 0000
XOR	0011 0000	1010 1011	1001 1011
g(w[7])	30	AB	9B

w[8] = g(w[7]) XOR w[4]

g(w[7])	0011 0000	1010 1011	1001 1011
w[4]	1100 1110	1010 0010	1101 1111
XOR	1111 1110	0000 1001	0100 0100
w[8]	FE	09	44

w[9] = w[8] XOR w[5]

w[8]	1111 1110	0000 1001	0100 0100
w[5]	1011 0111	1100 1101	1010 1010
XOR	0100 1001	1100 0100	1110 1110
w[9]	49	C4	EE

w[10] = w[9] XOR w[6]

w[9]	0100 1001	1100 0100	1110 1110
w[6]	1101 0100	1010 1100	1100 0100
XOR	1001 1101	0110 1000	0010 1010
w[10]	9D	68	2A

w[11] = w[10] XOR w[7]

w[10]	1001 1101	0110 1000	0010 1010
w[7]	1011 0000	1100 0011	1010 1101
XOR	0010 1101	1010 1011	1000 0111
w[11]	2D	AB	87

- Combine w[8], w[9], w[10], w[11] to get 2nd round key
- 2nd round key = **(FE, 09, 44, 19, 49, C4, EE, 64, 9D, 68, 2A, 39, 2D, AB, 87, 10)**

State matrix

w[8]	w[9]	w[10]	w[11]
FE	49	9D	2D
09	C4	68	AB
44	EE	2A	87
19	64	39	10

Circular byte left shift w[11] = (2D AB 87 10) » (AB 87 10 2D)

Substitution byte w[11] = (AB 87 10 2D) » (5C 29 F4 E6)

- Add round key constant (XOR operation)
- (5C 29 F4 E6) XOR (03 00 00 00)

Sub byte	0101 1100	0010 1001	1111 0100
Round 02	0000 0011	0000 0000	0000 0000
XOR	0101 1111	0010 1001	1111 0100
g(w[11])	5F	29	F4

w[12] = g(w[11]) XOR w[8]

g(w[11])	0101 1111	0010 1001	1111 0100
w[8]	1111 1110	0000 1001	0100 0100
XOR	1010 0001	0010 0000	1011 0000
w[12]	A1	20	B0

w[13] = w[12] XOR w[9]

w[12]	1010 0001	0010 0000	1011 0000
w[9]	0100 1001	1100 0100	1110 1110
XOR	1110 1000	1110 0100	0101 1110
w[13]	E8	E4	5E

w[14] = w[13] XOR w[10]

w[13]	1110 1000	1110 0100	0101 1110
w[10]	1001 1101	0110 1000	0010 1010
XOR	0111 0101	1000 1100	0111 0100
w[14]	75	9C	74

w[15] = w[14] XOR w[11]

w[14]	0111 0101	1000 1100	0111 0100
w[11]	0010 1101	1010 1011	1000 0111
XOR	0101 1000	0010 0111	1111 0011
w[15]	58	27	F3

- Combine w[12], w[13], w[14], w[15] to get 3rd round key
- 3rd round key = **(A1, 20, B0, FF, E8, E4, 5E, 9B, 75, 9C, 74, A2, 58, 27, F3, B2)**