Q1/ Write a complete program for encryption and decryption using ELGamal public key method.

Program in vb6

```vbnet
t1 = Second(Time())

beta = fast(alfa, k, p)
Print #2, beta

pb = fast(alfa, a, p)

zz = fast(pb, k, p)

While Not (EOF(1))
  xx = Asc(Input(1, #1))
  sekma = (xx * zz Mod p)
  Print #2, sekma;
Wend

t2 = Second(Time())

Text3.Text = t2 - t1

s = MsgBox("compelet encryption", 14, "Note")

End If
```
Q2/ Encrypt the message "computer" using the AES block cipher algorithm, the secret key is "science".

```
Q2/  
Sol1/ The plain text is: Computer  
The Key is: Science  
The ASCII for plain and key

C  99 01100111  
O  111 01101111  
M  101 01101101  
P  112 01100000  
u  117 01101111  
f  116 01101100  
e  101 01100101  
v  114 01101010

Convert the ASCII from decimal to hexadecimal

C ~ 99 = 63  
O ~ 111 = 6F  
M ~ 109 = 6D  
P ~ 112 = 70  
u ~ 117 = 75  
f ~ 116 = 74  
e ~ 101 = 65  
v ~ 114 = 72  
6 ~ 115 = 73  
c ~ 99 = 63  
i ~ 105 = 69  
e ~ 101 = 65  
r ~ 110 = 74  
c ~ 99 = 63  
e ~ 101 = 65  
# ~ 32 = 20

* The 12round 12o is ADD (plain, key)

63 6F 6D 70 + 73 63 69 65 6E 63 6F 20 = D6 ED D6 D5

The new transformation
```
The Round 2n (Substitute, Shift Row, and Key)

1. Substitute

<table>
<thead>
<tr>
<th>$D_6$</th>
<th>$E_6$</th>
<th>$D_5$</th>
<th>$D_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_6$</td>
<td>$E_3$</td>
<td>$E_6$</td>
<td>$D_3$</td>
</tr>
</tbody>
</table>

Each value as $D_6$ where $D$ is row and 6 is column in $S$-box:

- $D_6 = F_6$
- $E_6 = 55$
- $D_5 = F_6$
- $D_4 = 03$
- $E_3 = 11$
- $D_7 = 0E$
- $D_2 = 66$
- $D_3 = 4F$

Then:

<table>
<thead>
<tr>
<th>$F_6$</th>
<th>$55$</th>
<th>$F_6$</th>
<th>$03$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$11$</td>
<td>$0E$</td>
<td>$66$</td>
<td>$4F$</td>
</tr>
</tbody>
</table>

2. Shift Row:

- The Round Shift to left zero location
- The Round Shift to left 1 location

<table>
<thead>
<tr>
<th>$F_6$</th>
<th>$55$</th>
<th>$F_6$</th>
<th>$03$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$11$</td>
<td>$0E$</td>
<td>$66$</td>
<td>$4F$</td>
</tr>
</tbody>
</table>

Then the new transformation is:

<table>
<thead>
<tr>
<th>$F_6$</th>
<th>$55$</th>
<th>$F_6$</th>
<th>$03$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0E$</td>
<td>$66$</td>
<td>$4F$</td>
<td>$11$</td>
</tr>
</tbody>
</table>

3. Add Key

<table>
<thead>
<tr>
<th>$F_6$</th>
<th>$55$</th>
<th>$F_6$</th>
<th>$03$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0E$</td>
<td>$66$</td>
<td>$4F$</td>
<td>$11$</td>
</tr>
</tbody>
</table>

Then the new transformation is:

$$
\begin{bmatrix}
73 & 63 & 67 & 65 \\
6E & 63 & 6E & 20
\end{bmatrix} + \begin{bmatrix}
E2 & B8 & D8 & 68 \\
7C & C9 & BD & 31
\end{bmatrix} = \begin{bmatrix}
7C & C9 & BD & 31
\end{bmatrix}
$$

Then the new transformation is:

$$
E2 \ B8 \ D8 \ 68 \\
7C \ C9 \ BD \ 31
$$
Q3/Explain the following in detail

a/

<table>
<thead>
<tr>
<th>Q3/ Compare between the S-Box (DES, Blowfish)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DES</td>
</tr>
<tr>
<td>Blowfish</td>
</tr>
<tr>
<td>1 - 8 S-Box number</td>
</tr>
<tr>
<td>2 - input (6) and output (4)</td>
</tr>
<tr>
<td>3 - 2D</td>
</tr>
<tr>
<td>4 - 4 bits on each cell</td>
</tr>
<tr>
<td>5 - Replacement</td>
</tr>
<tr>
<td>6 - the X are 2-bit, Y 4-bit</td>
</tr>
<tr>
<td>7 - Static Value</td>
</tr>
<tr>
<td>8 - Full offline</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>1 - 4 S-Box number</td>
</tr>
<tr>
<td>2 - input (8) and output (32)</td>
</tr>
<tr>
<td>3 - 1-D</td>
</tr>
<tr>
<td>4 - S-Box array of 1-D, 32-bit cell</td>
</tr>
<tr>
<td>5 - Replacement</td>
</tr>
<tr>
<td>6 - 8-bit as index</td>
</tr>
<tr>
<td>7 - init to replace value</td>
</tr>
<tr>
<td>8 - Full on run program.</td>
</tr>
</tbody>
</table>
b/ Digital signature in RSA

The RSA public-key cryptosystem can be used for both encryption and signatures. Each user has three integers e, d and n, n = pq with p and q large primes. For the key pair (e, d), ed ≡ 1 (mod φ(n)) must be satisfied. If sender A wants to send signed message c corresponding to message m to receiver B, A signs it using A’s private key, computing

\[ c \equiv m^d \pmod{n_A}. \]

First A computes

\[ \phi(n_A) = lcm(p_A - 1, q_A - 1) \]

where \( lcm \) stands for the least common multiple. The sender A selects his own key pair \((e_A, d_A)\) such that

\[ e_A \cdot d_A \equiv 1 \pmod{\phi(n_A)} \]

The modulus \( n_A \) and the public key \( e_A \) are published. The following Figure illustrates the RSA signature scheme.

![RSA Signature Scheme Diagram](image-url)
**Example 1:**

Choose $p = 11$ and $q = 17$. Then $n = pq = 187$.

Compute $\phi(n) = 1cm(p-1,q-1) = 1cm(10,16) = 80$

Select $e_A = 27$. Then $e_Ad_A \equiv 1 \pmod{\phi(nA)}$

$$27d_A \equiv 1 \pmod{80}$$

$$d_A = 3$$

Suppose $m = 55$. Then the signed message is

$$c \equiv m^{d_A} \pmod{187}$$

$$\equiv 55^3 \pmod{187} \equiv 132$$

The message will be recreated as:

$$m \equiv c^{e_A} \pmod{n}$$

$$\equiv 132^{27} \pmod{187} \equiv 55$$

Thus, the message $m$ is accepted as authentic.

---

**c/Encryption algorithm for McEliece.**

1- Integers $K,N$ and $T$ are fixed as common system parameter.

2- Each entity $A$ should

2.1 Choose a $K*N$ generator ($G$) for a binary $(N,K)$ liner code which can correct $(T)$ errors and for which an efficient decoding algorithm is know.

2.2 Select a random $K*K$ binary non-singular matrix $S$
2.3 Select a random $N \times N$ permutation matrix $P$

3. Compute $K \times N$ matrix $\tilde{G} = SG$

4. Alice public key is $(\tilde{G}, T)$ and private key $(S, G, P)$

- **Algorithm: Encryption**
  Encryption Bob should do the following

  1. Obtain Bob authentic public key $(\tilde{G}, T)$
  2. Representation the message as binary string $M$ of length $K$.
  3. Choose a random binary vector $(Z)$ of the length $(N)$ have at most $(T)$
  4. Compute the binary vector $C = M \tilde{G} + Z$
  5. Send the cipher text $C$ to Alice

- **Algorithm: Decryption**

  To recover the plain text $(M)$ from $C$, Alice should do the following:

  1. Compute $\tilde{C} = CP^{-1}$, where $P^{-1}$ is the inverse of the matrix $P$
  2. Compute $M^{\wedge} = \tilde{C}[\tilde{C}_1, \tilde{C}_2, \ldots, \tilde{C}_n]$
    
    If $\tilde{C}_i > 0$ then $M^{\wedge}_i = 1$ else $M^{\wedge}_i = 0$

  3. Compute $M = M^{\wedge} S^{-1}$

**Q4/Rind the Result:**

1. 

$$\Phi(M^{\wedge}) = M^{\wedge - 1} (M - 1)$$

Find $\Phi(3^{\frac{1}{3}})$ the number 9 is not prime then:

$$\Phi(3^{\frac{1}{3}}) = 3^{\frac{1}{3}}(3-1) = 9 \times 2 = 18$$

$$\{2, 4, 5, 7, 8, 10, 11, 13, 14, 16, 17, 19, 20, 22, 23, 25, 26\}$$

2. $48 = 2 \times 2 \times 2 \times 2 \times 3$, 
180 = 2 \times 2 \times 3 \times 3 \times 5.

What they share in common is two "2"s and a "3"

Least common multiple = 2 \times 2 \times 2 \times 2 \times 3 \times 3 \times 5 = 720

Greatest common divisor = 2 \times 2 \times 3 = 12.

3 - (-4) \mod 7 = 3 \mod 7 = 3

4 \cdot (-5)^2 \mod 3

25 \mod 3 = 1
Q5/

1. Design $IP$ and $IP^{-1}$ for DES (16 input, 16 output). The array of input and output is $4 \times 4$ cell and the index (1 to 16).

\begin{center}
\begin{tabular}{ccc}
10 & 16 & 2 \\
8 & 13 & 6 & 14 \\
12 & 15 & 9 & 3 \\
11 & 1 & 4 & 7 \\
\end{tabular}
\end{center}

\begin{center}
\begin{tabular}{ccc}
14 & 3 & 12 & 15 \\
4 & 7 & 16 & 10 \\
11 & 1 & 13 & 9 \\
6 & 8 & 10 & 2 \\
\end{tabular}
\end{center}

Q5/

2. Can enhance and block.

Q6/

a/ The Feistel (F) Function

The F-function, depicted in Figure (2-2), operates on half a block (32 bits) at a time and consists of four stages:
1. **Expansion** the 32-bit half-block is expanded to 48 bits using the *expansion permutation*, denoted $E$ in the diagram, by duplicating some of the bits.

2. **Key mixing** the result is combined with a *subkey* using an XOR operation. Sixteen 48-bit subkeys, one for each round are derived from the main key using the *key schedule* (described below).

3. **Substitution** after mixing in the subkey, the block is divided into eight 6-bit pieces before processing by the *S-boxes*, or *substitution boxes*. Each of the eight S-boxes replaces its six input bits with four output bits according to a non-linear transformation, provided in the form of a look up table. The S-boxes provide the core of the security of DES without them; the cipher would be linear, and trivially breakable.

4. **Permutation** finally, the 32 outputs from the S-boxes are rearranged according to a fixed permutation, the *P-box*. The alternation of substitution from the S-boxes, and permutation of bits from the P-box and Expansion provides so-called "confusion and diffusion" respectively, a concept identified by Claude Shannon in the 1940s as a necessary condition for a secure yet practical cipher.

b/ The subkeys are calculated using the **Blowfish algorithm**. The exact method follows.
(1) Initialize first the P-array and then the four S-boxes, in order, with a fixed string. This string consists of the hexadecimal digits of p.

(2) XOR $P_1$ with the first 32 bits of the key, XOR $P_2$ with the second 32-bits of the key, and so on for all bits of the key (up to $P_{18}$). Repeatedly cycle through the key bits until the entire P-array has been XORed with key bits.

(3) Encrypt the all-zero string with the Blowfish algorithm, using the subkeys described in steps (1) and (2).

(4) Replace $P_1$ and $P_2$ with the output of step (3).

(5) Encrypt the output of step (3) using the Blowfish algorithm with the modified subkeys.

(6) Replace $P_3$ and $P_4$ with the output of step (5).

(7) Continue the process, replacing all elements of the P-array, and then all four S-boxes in order, with the output of the continuously changing Blowfish algorithm.

In total, 521 iterations are required to generate all required subkeys. Applications can store the subkeys—their’s no need to execute this derivation process multiple times.

**NOTE:**

P=18 32-bit to byte (div 8)

\[ P=18\times4=72 \]

S=256 32-bit to byte (div 8)

\[ S=256\times4\text{ (byte)}\times4\text{ (number s-box)}=4096 \]

The total= 4168

In each round we replace 2 p then 8 round to replace all p

Then

\[ 521\text{ (number Round)}\times8\text{ (Number P-)}=4168 \text{ byte as subkey} \]