Q3A-3:
In the ECB mode any identical plaintexts will encrypt to the same ciphertext value. Unless each plaintext is encrypted only once within the lifetime of the key, you are revealing information about your plaintexts. ECB mode is not ideal for any cryptosystem which actually desires to keep its ciphertexts secure. ECB mode is totally insecure. The same block in the plaintext results in the same ciphertext. That means that it's deterministic and the attacker can distinguish between two ciphertexts so it's not CPA secure.

Suppose the attacker is giving the challenger 2 equal blocks of messages m0||m1. He will receive c0||c1 where c0=c1. And then he submits another message of two blocks m2||m3 where m2<>m3 along with the first message m0||m1. Since he can distinguish between those two the scheme is insecure under Chosen Plaintext Attacks.

Q3B: CBC encryption is best utilized when the encrypted data is completely diffused with the help of a completely random IV each time. If the attacker can predict the IV you are going to use to encrypt the message, your cipher is no longer CPA secure. As an example:

Round 1:
IV = 100
IV_binary = 00000000 01100100

plaintext = "The plain text!"
plaintext_binary = 01010100 01101000 01100101 00100000 01110000 01101100 01100001 01101001 01101110 00100000 01110100 01100101 01111000 01110100 00100001

IV_binary XOR plaintext_binary

```
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 01100100
01010100 01101000 01100101 00100000 01110000 01101100 01100001 01101001 01101110 00100000 01110100 01100101 01111000 01110100 00100001
--------------------------------------------------------------------------------------------------------------------------------------
01010100 01101000 01100101 00100000 01110000 01101100 01100001 01101001 01101110 00100000 01110100 01100101 01111000 01110100 01000101
```

ivbin_xor_plaintext = 01010100 01101000 01100101 00100000 01110000 01101100 01100001 01101001 01101110 00100000 01110100 01100101 01111000 01110100 01000101
decoded string (xor_ivbin_plaintext) = "The plain textE"

Round 2:
IV = IV+1 = 101
IV_binary = 00000000 01100101

desired_plaintext = IV_binary XOR missing_plaintext
desired_plaintext_binary = 01010100 01101000 01100101 00100000 01110000 01101100 01100001 01101001 01101110 00100000 01110100 01100101 01111000 01110100 00100001

IV_binary XOR missing_plaintext_finder

```
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 01100101
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 01000100
--------------------------------------------------------------------------------------------------------------------------------------
01010100 01101000 01100101 00100000 01110000 01101100 01100001 01101001 01101110 00100000 01110100 01100101 01111000 01110100 00100001
```

Missing_plain_text(m2) = 01010100 01101000 01100101 00100000 01110000 01101100 01100001 01101001 01101110 00100000 01110100 01100101 01111000 01110100 01000100
Decode string(m2) = "The plain textD"

Answer:
message1 = "**The plain text!**" with IV 100
message2 = "**The plain textD**" with IV 101

You must use a cryptographically **random IV** of the same block size as the cipher (AES-256 uses a 128-bit block size). Use of a constant IV is essentially indistinguishable from ECB mode, and use of weak, predictable IVs isn't much better either.

Q3C: OFB
It is possible to recover the corresponding O_i block

$O_i = C_i \oplus IV_{i+1}$

Q4(a)
i – Calculate Φ(n) when e=3, gcd(e, Φ(n))≠1, p = 5, q = 13
Ans:
Φ(n) = (p-1) (q-1)
Φ(n) = 4 x 12
Φ(n) = 48

ii – Encrypt 2 and 57
Ans:
Encrypting 2
C = m^e mod N
C = $(2)^3$ mod (p x q)
C = 8 mod 65
C = 8
Encryption for 2 is 8

Encrypting 57
C = m^e mod N
C = $(57)^3$ mod (p x q)
C = 185193 mod 65
C = 8
Encryption for 57 is also 8

iii – Find decryption key (d) such that ed ≡ 1 mod Φ(n)
Ans:
d = (k Φ(n) + 1) / e
d = (k (48) + 1) / 3
d = (48k + 1) / 3
Since **gcd(e, Φ(n)) ≠ 1**, no matter what value we put in the above equation, it will not return an integer value for d, which is why we are not able to find the decryption key. On the other hand, encryption values for 2 and 57 both are same.

Q4(b) Decrypt the corresponding plain text when C = 10, e = 5 and N = 35.
Ans:
N = 35 = 5 * 7
Φ(n) = (5-1) (7-1) = 24
d = (k Φ(n) + 1) / e
d = (k (24) + 1) / 5
d = ((1 x 24) + 1) / 5
d = 25 / 5
d = 5

Decryption key = 5

Q5A-2: Public key should be secured. Encrypted ciphers can be hashed before transmission.