

### Problem 1

- ☒ Compress then encrypt.
- ☒ Encrypt then compress.
- ☒ The order does not matter – either one is fine.
- ☐ The order does not matter – neither one will compress the data.

A:

原理上來說，壓縮演算法使用的是資料中的統計冗餘（如自然語言或許多檔案格式中存在的冗餘）來進行壓縮，但加密的副作用就是會一併消除這些統計冗餘，因此加密資訊通常無法很好地壓縮。

目前絕大部分作法都是先壓縮再加密，因為通常加密演算法針對較短的資料會有更好的表現。

然而教授有在上課提及目前也有先加密再壓縮的技術，雖然我目前並沒有查詢到相關實例。

—

### Problem 2

- ☐  $G'(k) = G(k) \parallel G(k)$
- ☒  $G'(k) = G(k \oplus 1^s)$
- ☐  $G'(k) = G(0)$
- ☐  $G'(k) = G(1)$
- ☐  $G'(k) = G(k) \parallel 0$
- ☒  $G'(k_1, k_2) = G(k_1) \parallel G(k_2)$
- ☒  $G'(k) = \text{reverse}(G(k))$
- ☒  $G'(k) = \text{rotation}_n(G(k))$

A:

1.  $G'(k) = G(k) \parallel G(k)$ : 因為  $G$  的值域固定為  $n$  bits 的數字，非常容易看出每  $n$  個字符會開始重複，因此不能作為 secure PRG。
3.  $G'(k) = G(0)$ : 並無隨機性，因此不能作為 secure PRG。
4.  $G'(k) = G(1)$ : 並無隨機性，因此不能作為 secure PRG。
5.  $G'(k) = G(k) \parallel 0$ : 攻擊者可以確定最後一個 bit 必定為 0，因為在最後一個 bit 上違反 secure PRG 的定義，因此不能作為 secure PRG。

—

### Problem 3

- ☐  $p_1 = (k_1, k_2), p_2 = (k_1, k_2), p_3 = (k_2')$
- ☐  $p_1 = (k_1, k_2), p_2 = (k_1', k_2'), p_3 = (k_2')$
- ☒  $p_1 = (k_1, k_2), p_2 = (k_1', k_2), p_3 = (k_2')$
- ☐  $p_1 = (k_1, k_2), p_2 = (k_2, k_2'), p_3 = (k_2')$
- ☐  $p_1 = (k_1, k_2), p_2 = (k_1'), p_3 = (k_2')$

A:

依照題目提供的條件：

已知  $k_1 \oplus k_1' = k_2 \oplus k_2' = k$

- 1.任兩組pairs可以還原k,
- 2.單一個pair不可能還原k。

只有選項3的組合可以滿足所有要求：

$$p1 \& p2: k1 \oplus k1' = k, \quad k2 \oplus k2' = k$$

$$p1 \& p3: k2 \oplus k2' = k$$

$$p2 \& p3: k2 \oplus k2' = k$$

選項1, 2, 5不滿足條件1, 選項4不滿足條件2。

—

#### Problem 4

- ☐ No, there is a simple attack on this cipher.
- ☒ Yes
- ☐ No, only the One Time Pad has perfect secrecy.

A:

perfect secrecy的定義為：

$$\Pr[E(k, m_0) = c] = \Pr[E(k, m_1) = c]$$

其中  $m_0, m_1$  為  $M$  中任意兩個元素； $c$  為  $C$  中任意一個元素； $k$  為  $K$  中任意一個元素。

而在本題題目中對  $M$  與  $C$  中任意的各一個元素，都能滿足  $\Pr[E(k, m) = c] = 1/256$ ，因此符合 perfect secrecy。

—

#### Problem 5

- ☐  $E'(k, m) = E(0^n, m)$
- ☒  $E'((k, k'), m) = E(k, m) \parallel E(k', m)$
- ☐  $E'(k, m) = E(k, m) \parallel \text{MSB}(m)$
- ☒  $E'(k, m) = 0 \parallel E(k, m)$  (i.e. prepend 0 to the ciphertext)
- ☐  $E'(k, m) = E(k, m) \parallel k$
- ☒  $E'(k, m) = \text{reverse}(E(k, m))$
- ☒  $E'(k, m) = \text{rotation}_n(E(k, m))$

A:

1.  $E'(k, m) = E(0^n, m)$ : 由於加密的密鑰是固定的，所以對於任意的兩個不同的訊息  $m_1$  和  $m_2$ ，它們的加密結果  $E(0^n, m_1)$  和  $E(0^n, m_2)$  將是相同的，因此不能作為 semantically secure。

3.  $E'(k, m) = E(k, m) \parallel \text{MSB}(m)$ : Most Significant Bit 的值對於每個訊息都是固定的，可能會被發現最後一個 bit 正是明文的 Most Significant Bit，某種程度上使明文更可能被破解，因此不能作為 semantically secure。

5.  $E'(k, m) = E(k, m) \parallel k$ : 由於已知明文的空間和密鑰的空間，在進行足夠多的試驗後便會發現，無論對任何明文輸入給定  $k$ ，輸出都一定會洩漏  $k$  的資訊，之後可從各項輸出推論出明文，因此不能作為 semantically secure。

## Problem 6

A:


6962c720079b8c86981bc89a994d


只要先把C1轉換成bytes再與encode成UTF-8的P1做XOR得出OTP key  
再用該OTP key與encode成UTF-8的P2做XOR得到C2\_bytes  
把C2\_bytes轉成16進制即可得到題目所要的C2

```
1  P1 = 'attack at dawn'
2  C1 = '6c73d5240a948c86981bc294814d'
3  P2 = 'defend at noon'
4
5  C1_bytes = bytes.fromhex(C1)
6
7  OTP = bytes([a^b for a,b in zip(C1_bytes,P1.encode())])
8  C2_bytes = bytes([a^b for a,b in zip(OTP,P2.encode())])
9
10 C2 = C2_bytes.hex()
11 print(C2)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\MSI PC\Desktop\Dummy_Device_IoTtalk_v1_py-master
> python -u "c:\Users\MSI PC\Desktop\109705002.py"
6962c720079b8c86981bc89a994d
```

 Code

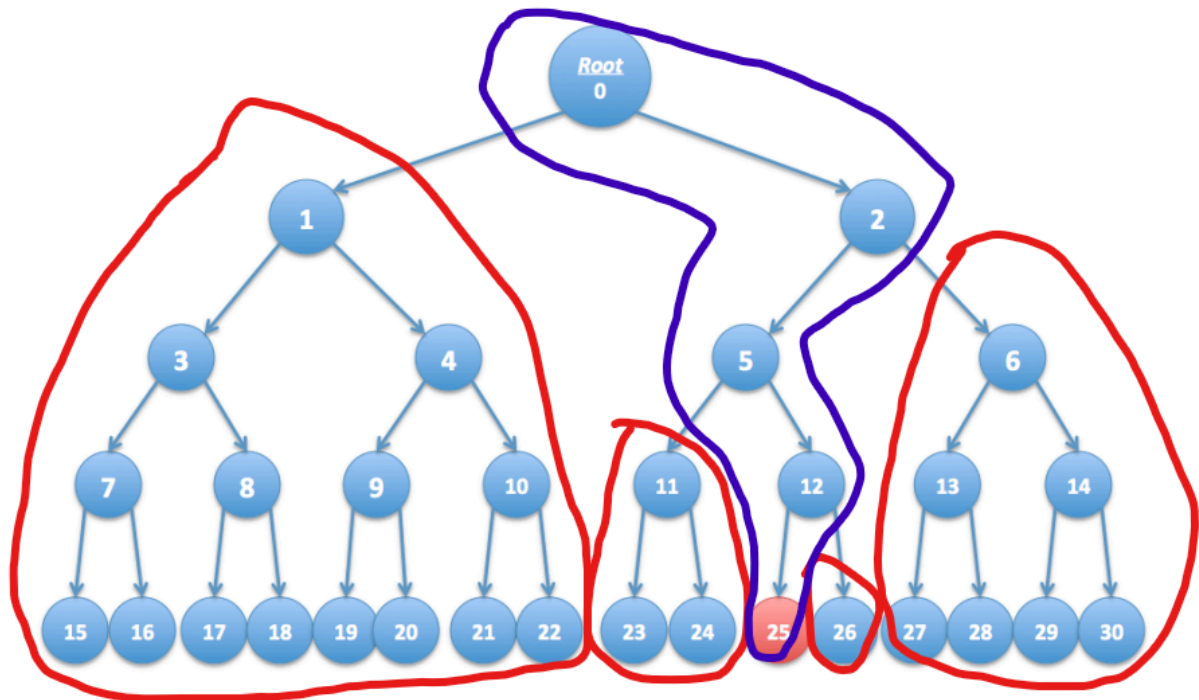
 Code

## Problem 7

- ☐ 21
- ☐ 17
- ☐ 5
- ☒ 26
- ☒ 6

- ☒ 1
- ☒ 11
- ☐ 24

由於不能讓25成功decrypt, 所以其路徑: 0 -> 2 -> 5 -> 12 -> 25皆不可以選來encrypt, 於是我們只選用1, 6, 11, 26來讓其他的player皆可以decrypt。



### Extra Credit

SHA-256與SHA-512-truncated-to-256-bits(截斷至 256 位的 SHA-512)在許多方面相似但無法提供相同的security properties。通常來說SHA-256 是更好的選擇

理由經過ChatGPT提供並被我整理有三大點：

**安全保證：**SHA-256 是非常流行的Hash函式，已經受到密碼學家的廣泛研究和分析，並且經受住了嚴格的檢驗。目前被廣泛採納為各種密碼學應用的Hash函式。

**標準化和兼容性：**SHA-256 是一個標準Hash函數，在各種加密庫、協議和系統中得到了廣泛支持。它更常用且在不同平台之間更具互通性。

**效能：**SHA-256的輸出大小是 256 位元，而雖然 SHA-512-truncated-to-256-bits 的輸出大小也是 256 位元，但 SHA-512 需要更多的計算資源來計算完整的 512 位元哈希值然後再截斷，計算量較小也是SHA-256的優勢之一。