1. Shor's Algorithm can be used to do which of the following?

1. Create superposition
2. Create entanglement
3. Factor very large integers
4. Increase algorithm time

Ans. 3

2. Why would RSA encryption be considered unsafe from quantum algorithms?

1. It includes quantum-proof factoring
2. Its factors can be determined using Shor's Algorithm
3. It is an outdated technology, even for traditional computers
4. The key is solved in polynomial time

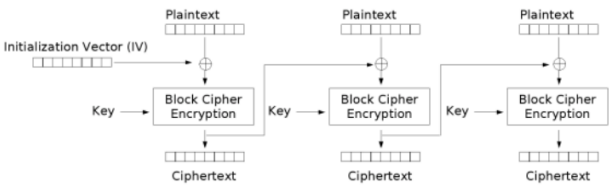
Ans. 2, 4

3. Why is AES-GCM preferred and the AES-CBC support was removed in TLS1.3 ?

Ans. 因為有攻擊方法可以將明文解密出來，像是老師上課提到的 Beast Attack。

The Beast Attack

- CBC 模式就是把前一塊密文 C0 和這一塊要加密的明文 P1 先 XOR，然後再去加密，所以可以看作以下函式：
- **C1 = Encrypt(Key, C0 ⊕ P1)**
- 而一開始的 C0 因為沒有東西可以跟它 XOR，所以演算法會產生一組 Initialization Vector (IV)，長度與 block 一致，然後 C0 = Encrypt(Key, **IV ⊕ P0)**



Cipher Block Chaining (CBC) mode encryption

- CBC 模式。因此你可以想像，透過 CBC 模式，即使明文中有兩處一模一樣的 block，加密完後它們也會長得不同。每份明文都會有一個 IV，然後透過 CBC 一直串下去，產生出一整份的密文。
- 對 HTTP 來說，關鍵在於怎樣算是「一份明文」呢？因為 HTTP 協定是 stateless 的協定，也就是每次 client 送一個 request，server 回應一個 response。
- **SSL/TLS 的規格中因此定義了 Record 的概念**，把應用軟體所丟出來的資料切割成**一個又一個的 Record，在表頭定義 Record 的長度，然後每一個 Record 就看作一份明文。**
- 使用 CBC有兩種可能：
  1. 每一個 Record 都是獨立的，一個 Record 就產生一個新的 IV。
  2. **全部的 Record 都使用同一個 IV，不斷地從前面 Record 累積下來的密文串接下去。**

- **而有問題的 SSL/TLS 版本，就是使用第二種方式**。所以只要攻擊者可以取得 Ci-1 和 Ci，就有辦法可以推測出 Pi，只是這辦法相當難，它是這樣運作的：
  假設攻擊者可以作到下面兩件事：
  1. 偷看每一個 C (取得 Ci-1 和 Ci) 。這不難作到，透過 Sniffer 就可以。
- 2. 操控 Client 傳送出某些特製的資料。難，而 BEAST 就是在這方面努力。要操控 HTTP 沒想像中困難，例如只要透過一個 <img src="..."> 就可以控制 client 送出某些你安排好的 request，但是要很有彈性的操控，例如指定要 client 送出某些 bytes 就沒那麼容易，透過 applet 或是 wesocket 可以作到 。**但是麻煩是 Same Origin Policy (SOP) 讓 client 端的 applet 或是 websocket 都只能夠在同一個網站下執行，除非被跨的那個網站有特別允許。**

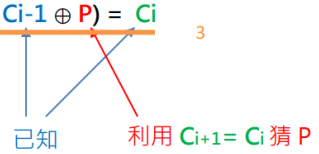
假設我們可以作到以上兩件事，模擬情況是我們已經**觀測到 Ci-1 和 Ci 這兩組已知密文**，但是我們不知道 Pi-1 或 Pi，**現在我們的目標是求得 Pi**，所以我們**先窮舉一個 P，並假設它等於 Pi**。

再來，我們操控 client 傳送出 **Ci ⊕ Ci-1 ⊕ P** 的資料，當 SSL/TLS 加密這塊資料時，就會變成以下：

$Ci+1 = Encrypt(Key, Ci \oplus (Ci \oplus Ci\text{-}1 \oplus P)) = Encrypt(Key, Ci\text{-}1 \oplus P) = Ci$

1 2 3

$C_{n+1} = Encrypt(Key, C_n \oplus P_{n+1})$



如果我們窮舉出的那個 P 真的等於真的 Pi 的話，那麼 Ci+1 就會等於 Ci。反過來說，**我們只要觀察 Ci+1 是否等於 Ci，就可以知道我們窮舉出的 P 是否正確**。只要透過不斷地窮舉，我們就可以找出真正的 Pi。

因此，這個攻擊法並不是找出 SSL/TLS 的金鑰KEY，而是解密出其中某塊明文。

而 GCM 是利用 CTR mode encryption，不會有 CBC 重複利用前面密文的問題。

GCM: CTR mode encryption then CW-MAC