1-1.

for every m,c if E(k, m) = c

E(k, m) = mk mod p = c ==> ck^-1 = mkk^-1 mod p = m

而在Zp中k的乘法反元素k^-1只有一個，所以此方法是perfect secrecy

1-2.

for every m,c if E(k, m) = c

E(k, m) = k xor m = c   ==> k = m xor c

k = m xor c算出來的k只有一個，所以OTP是perfect secrecy

1-3.

不會受影響，因為OTP的key是uniform的取的，而和unifrom的key做xor後的cipher也會uniform，所以不會從cipher中透漏任何統計的資訊

1-4.

因為attacker也可以使用公鑰加密，他可以送出一個訊息透過公鑰加密後，一直測試不同的密鑰直到能夠正確解開，這樣他就得到真正的密鑰可以解出任何透過此公鑰加密的訊息了。

這違反了「獲取cipher不會提供任何plaintext的資訊」，所以他不是perfect secrecy

2-1.

因為知道p和xi~xj，我們可以列出很多mod的方程式，只要夠多我們就可以解出a和b，即可透過生成的式子預測之後的sequence

2-2.

不適合被拿來生成key來加密東西，因為就算a,b,p都不知道，他仍然很容易被破解

2-3.

1，知道一個即可透過生成的公式推出剩下的值了

2-4.

3，因為現在只有a, b兩個變數不知道，所以可列出兩個方程式即可解出a, b

3.

$$\oplus \equiv \bar{p}q + p\bar{q}$$

$$\overline{(x_1 x_2)}\,(\bar{x}_1 x_3) + (x_1 x_2)\,\overline{(\bar{x}_1 x_3)}$$

$$= (\bar{x}_1 + \bar{x}_2)(\bar{x}_1 x_3) + (x_1 x_2)(x_1 + \bar{x}_3)$$

$$= \bar{x}_1 x_3 + \bar{x}_1 \bar{x}_2 x_3 + x_1 x_2 + x_1 x_2 \bar{x}_3$$

$$= \bar{x}_1 x_3 (1 + \bar{x}_2) + x_1 x_2 (1 + \bar{x}_3) = \bar{x}_1 x_3 + x_1 x_2$$

$x_3 = 0$

x_1 \ x_2	0	1
0	0	0
1	0	1

$x_3 = 1$

x_1 \ x_2	0	1
0	1	1
1	0	1

$$\Rightarrow \Pr\left(G(t) = x_2(t)\right) = \frac{3}{4}$$

4.

Alice要證知道x值(就是密碼)

1. Alice與system約定一個質數P、一個在Zp的乘法生成元g
2. Alice計算y = g^x mod p，傳給system。
3. 重複以下步驟：
    1. Alice 從uniform分布的地方隨機選一個數字r，計算C = g^r mod p，傳送C給system。
    2. system問Alice (x+r) mod (p-1) 或問r

       2-1. 若問(x+r) mod (p-1)，則system驗證 (C*y) mod p = g^((x+r) mod (p - 1)) mod p

       2-2. 若問r，則system驗證C = g^r mod p

(x+r) mod (p-1)可視為x mod (p-1)的加密；若r uniform，(x+r) mod (p - 1)也同樣會unifrom。所以不會洩漏x的任何資訊。

5.

用三個LFSR組成一個Alternating step generator(ASG)，他的輸出是兩個LFSR的XOR，而第三個LFSR用來為這兩個LFSR交替提供clock，也就是如果第三個LFSR輸出0的話，則為第一個LFSR提供clock使他往前一步；反之第三個LFSR輸出1的話，則為第二個LFSR提供clock使他往前一步。

這三個LFSR會用不同但degree接近的primitive polynomials，且初始值非0，令他們三個都是可以輸出產生最多不同值的LFSR。

6-1.

因為gcd(a, 26) = 1，a有12種可能；1<=b<=26，b有26種可能，因此Affine Ciphers的key sapce是12*26=312 (假設完全沒變也可以)

6-2.

26!，因為每個字母可map到全部目前還沒被map到過的字母 (假設完全沒變也可以)

6-3.

26，因為在mod 26的情況下，可以shift 0~25而不會重複 (假設完全沒變也可以)

7-1.

P0 = m0

C0 = fk(0 xor m0) = fk(m0) = T0

P1 = m0,m1

C0 = fk(0 xor m0) = fk(m0)

C1 = fk(C0 xor m1) = fk(fk(m0) xor m1) = T1

而我們可以令 P2 = m0,m1,(m0 xor T1)

C0 = fk(0 xor m0) = fk(m0)

C1 = fk(C0 xor m1) = fk(fk(m0) xor m1) = T1

C2 = fk(C1 xor m2) = fk(C1 xor m0 xor T1) = fk(m0) = T0 = T2 (因為C1 = T1)

如此我們便造出了新的message，不是原本的P0，卻產生了和P0一樣的tag，這樣就會有問題，所以才不支援variable length input

7-2.

Alice 將plaintext = P0, P1, ..., Pn-1, Pn經過Ek加密為cipher = C0, C1, ..., Cn-1, Cn 傳給Bob，此cipher對應的tag為t = Cn

若Eve截取了此cipher，把他改成cipher' = C0', C1', ..., Cn-1', Cn給Bob，其中C0~Cn-1皆可被竄改，而Cn需與原本相同

則此cipher'解密出來的plaintext' = P0', P1', ..., Pn-1', Pn'已與原本的plaintext不同，而其中Pn' = Cn-1' xor Ek^-1(Cn)

此cipher'對應的tag t' = Ek(Pn' xor Cn-1') = Ek(Cn-1' xor Ek^-1(Cn) xor Cn-1') = Ek(Ek^-1(Cn)) = Cn = t

也就是cipher被Eve更改後，且對應的plaintext也被改掉，而Bob卻仍認為此訊息就是Alice傳了一個Alice沒有傳的訊息(因為和原本Alice傳的不同了)

8.

程式在檔案0816147.py

關鍵1.若對slience的部分做shift，然後和原本的做xor，原本的plaintext的部分就都會變1，再對結果取nor運算，即可得到k'[i] = k[i] XOR k[i+1]

關鍵2.假設一段對話中會有很多silence的部分，所以對每段2*n都假設他是silence，取出來算Berlekamp_Massey，統計哪個poly出現最多次代表他有可能是真正對應的poly

關鍵3.我們知道哪邊是silence的，XOR完取出來的亂數應該也是k'[i] = k[i] XOR k[i+1]，所以就假設k[i]的開頭為0或1，並根據假設，利用xor往後推出k[i]後面的值，如此便可產生2種可能的原本key，再對這兩個key都去做Berlekamp_Massey

流程：

1.讀cipher

2.shift並做xor (根據關鍵1)

3.對xor後的cipher取很多小段，假設他是silence，找出這段可能對應的兩種key，分別對他們做Berlekamp_Massey來求出他對應的多項式，統計哪個多項式被對應到最多次 (根據關鍵2 3)

4.利用多項式與seed去生成指定長度的key

5.算出plaintext

6.輸出到檔案