Problem 1

1. In this case the set of plaintexts, ciphertexts and keys all coincide with Z_p^*.

In this case the criteria to decide if a cipher E has perfect secrecy are the following:

(1). the key must be chosen at random uniformly

(2). for every plaintext m and every ciphertext c, there exists a unique key k such that $E_k(m) = c$.

The text of the problem already said that keys were chosen at random uniformly, thus we need to check only condition (2).

Fix a plaintext m and a ciphertext c in Z_p^*.

Since the encyrption rule is $c = mk \bmod p$, we know that the key that maps m into c is uniquely determined as $k = cm^{-1} \bmod p$.

Since Z_p^* is a group inverses exist and are unique also $cm^{-1} \bmod p = mm^{-1}c = k \bmod p$

$k = cm^{-1} \bmod p$

2. 一個 cipher (E, D) over (K, M, C) 滿足以下條件時，擁有 perfect secrecy:

(1). 對每一組明文 m0, m1 in M (length(m0) = length(m1)) 和每一個密文 c in C 來說，Pr[E(k, m0) = c] = Pr[E(k, m1) = c]

(2). Pr[…] 表示機率函式，k 是 K 裡隨機選中的數 (uniform distribution)

One time pad 會隨機選出和明文 m 一樣長的密鑰 k，

對 E(k, m) = k xor m 來說，若 k 是 independent uniform random variable，

k xor m 的結果也會是 uniform random variable。

uniform random variable 顯然滿足 perfect secrecy，套回 perfect secrecy 的定義，給定 m0、m1、c，滿足 E(k, m0) 和 E(k, m1) 的 k 各別只有一個，所以攻擊者無法從密文推測出用哪個明文加密。

3. Not susceptible to statistical analysis

4. No.

The following attack will work against any public-key encryption scheme:

(1). The attacker has access to the public key and, say, a ciphertext encrypting some unknown message

(2). for every possible secret key (by enumerating the key space), the attacker does the following:

(i). checking whether the secret key does correspond to the known public key (for instance, by encrypting every single possible message then trying to decrypt them using this secret key)

(ii). when a secret key was found that passed the test of the previous step, the attacker uses it to decrypt the message

Of course, this attack will take a gigantic amount of resources, but it will work against any public-key encryption scheme. Now this goes against the idea of perfect secrecy which says that getting a ciphertext gives no information about the plaintext, even for an adversary with unbounded resources.

Problem 2

1. Assume the attacker sees only $S = x_0,\ x_1,\ x_2$.

This gives him two linear equations in two unknowns (the coefficients a, b of the generator). That is $ax_0 + b = x_1 \text{ and } ax_1 + b = x_2$

The determinant of the above system is $x_0 - x_1$. Thus if $x_0 - x_1 \neq 0$ the attacker can solve the system of equations and determine a, b and keep predicting the sequence applying the rule $x_i = ax_{i-1} + b \bmod p$.

If $x_0 - x_1 = 0$, then $x_0 = x_1$ and the sequence is clearly constant, i.e. $x_i = x_0$ for all i.

2. Although this was a simplified version of the linear congruential generator (the attacker knew the modulus p used), this behavior tells us that linear congruential generators are not secure for cryptographic applications.

3. One bit. Since we know one bit, we can predict the next bit recursively by the rule of $x_i = ax_{i-1} + b \ mod \ p$

4. Three bits. Since if less than three bits, i.e. we have one bit or two bits, we can only list one sentence of polynomial or less. This can't solve two unknown values by knowing one sentence of polynomial.

Problem 3

$$Prob[G(t) = x_2(t)] = Prob[x_1(t) = 1] + Prob[x_1(t) = 0] \cdot Prob[x_3(t) = x_2(t)] = \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{2} = \frac{3}{4}$$

Problem 4

There are several ways of doing this. The simplest one is for the server to display a random "challenge" R at each login. Alice will then type in $C = P \oplus R$ where P is the password. Since the pad R is random and different at each time the attacker does not learn anything about the password.

Of course, any unique transformation would do. For example, assume that the password can be any 3-digit number (i.e. a number between 000 and 999).

Suppose Alice's password is P = 513. Then she can type in $C = R + P \bmod 1000$. If the challenge is, say, 618 then C = 1131 = 131 mod 1000.

To avoid the need for Alice to do math in her head or to use a calculator, a simpler protocol can be established. For example, Alice may be requested to “fix” each digit by going “up” or "down” with the arrow keys on the keyboard. In the previous example she would hit the "down” key once on the first digit and 5 times on the last digit and then she would see the password on the screen and press enter.

Problem 5

用 3 個 LFSR，

$LFSR_0$ 決定 $LFSR_1$, $LFSR_2$ 哪個要進下一個 step

Output 為 $LFSR_1$ $\boldsymbol{XOR}$ $LFSR_2$

Pseudocode:

if ($LFSR_0$ == 1)

then $LFSR_1$. next()

else

then $LFSR_2$. next()

output = $LFSR_1$ $\boldsymbol{XOR}$ $LFSR_2$

Problem 6

1. $\varphi(26) = \varphi(2*13) = (2-1)*(13-1) = 12$, hence we have 12*26 = 312 possible keys.
2. 26!
3. 26

Problem 7

(a) CBC-MAC is not secure for variable-length messages. This is because an attacker who knows the correct message-tag pairs for two messages (m , t) and (m', t') can generate a third message m'' whose CBC-MAC will also be t'.

This is simply done by XORing the first block of m' with t and then concatenating m with this modified m'; i.e., by making $m'' = m \parallel [(m'_1 \oplus \mathrm{t}) \parallel m'_2 \parallel \cdots \parallel m'_x]$.

When computing the MAC for the message m'', it follows that we compute the MAC for m in the usual manner as t, but when this value is chained forwards to the stage computing $E_{k_{MAC}}(m'_1 \oplus \mathrm{t})$, we will perform an XOR operation with the value derived for the MAC of the first message. The presence of that tag in the new message means it will cancel, leaving no contribution to the MAC from the blocks of plain text in the first message m:

$E_{k_{MAC}}(m'_1 \oplus \mathrm{t} \oplus \mathrm{t}) = E_{k_{MAC}}(m'_1)$ and thus the tag for m'' is t'.

(b) Let $t1$ be the tag of message $m1$ = AAA4B6. We note that the length of message $'6'$ has been appended to it.

Similarly let $t2$ be the tag of message AAA4, and $t3$ be the tag of message CCC4.

Let $D := t2 \oplus t3$. Also, $E := B \oplus D = B \oplus t2 \oplus t3$

We now output $t1$ as a valid tag for message m' = CCC4E6. Clearly, while calculating CBC-MAC value of m', we get intermediate tag value, $t3$ for CCC4 which further evaluates to $= t3 \oplus E = t3 \oplus B \oplus t2 \oplus t3 = t2 \oplus B$, which appended with message length, 6 equates to $t2 \oplus B6$ that yield same tag value as $t1$.

Thus, a forged valid message tag pair can be generated for such scheme. However, such attack cannot be done if message length, $|m|$ is prepended with the message. This scheme has a drawback that the message length needs to be known apriori.

Problem 8