Problem 1

1. Confidentiality and integrity. We want only the authorized people to view and modify these keys for communication. (ASK UDAY integrity)
2. The protocol does not satisfy both security properties since if you can get *x* and *y* one can derive the secret key. So, it does not satisfy confidentiality property.

Problem 2

1.

{A,B,…,Z} turns into {0,1,…25}. So p_0 turns into 0123 and p_1 1436.

T=1,

If the period is one, then the key does not really do anything and changes the values by 1. Therefore, So the two passwords are disjoint, and you can calculate the correct password.

T=2,

Here the sets of possible encryptions are C0 = {(k1, k2 + 1, k1 + 2, k2 + 3) | k1, k2 ∈ Σ} and C1 = {(k1 + 1, k2 + 4, k1 + 3, k2 + 6) | k1, k2 ∈ Σ}. It is easy to see that C0 = C1: let (k1, k2 + 1, k1 + 2, k2 + 3) ∈ C_0 for some fixed k1, k2 ∈ Σ. Then it is also in C_1, because k 0 1 = k1 + 1 and k 0 2 = k2 + 3 are in Σ. The converse is also true. In other words, for every possible encryption c of p0 under key k there is a key k 0 that would have encrypted p_1 to c. Since k and k 0 are equally likely (since they are chosen from the uniform distribution over Σ × Σ), the adversary has no way of knowing which password was encrypted.

T=3,

This is the same problem as period 1 since the two passwords do not overlap with t=3.

T=4,

This is pretty much a onetime pad encryption since the length match up and therefore not hackable.

2.

Mono-alphabetic ciphers are easy to break with a chosen plaintext attack since you can find the one to one corresponding values of each key. You do not even need to brute force it. You can choose a sentence that contains all letters in the alphabet, and you have all encodings. To answer the last two questions, you need 25 unique letters to recover the entire key without having to guess. There are 26 letters and if you have 25 of the encodings you can deduce the 26^{th} letter quite easily. So 25 letters would also be the minimum needed to crack the encoding properly. If you had 20 letters however you can sort of guess what the last few letters are if you have enough ciphertext and are translating into plain English. This encryption would only work securely against an attacker if you had a small enough message that there could be multiple meanings for the message that you are trying to encode. (Anything less than 25 unique chars technically)

Problem 3

for this I used crib dragging, this method starts with simple words and common phrases and XORs the messages with this key. If words show up during the XOR it means that part of the message has some of the keyword in it. I did this with the Keyword Nikos and worked my way up to an entire key using the first few messages. Then I could print it all out using one message as my key.

```
python3 cribdrag.py
testing testing can you read this
yep I can read you perfectly fine
awesome one time pad is working
yay we can make fun of Nikos now
i hope no student can read this
that would be quite embarrassing
luckily OTP is perfectly secure
didnt Nikos say there was a catch
maybe but I didnt pay attention
we should really listen to Nikos
nah we are doing fine without him
```

See python program attached.