- CaesarCipher.py has 26 distinct keys - Number of Uppercase alphabets (26)

- a1p1.py has 52 distinct keys - Number of Uppercase alphabets (26) + number of Lowercase alphabet (26)

- a1p2.py also has 52 distinct keys because if the first letter is correctly deciphered, so will the rest be as the next letter depends on the previous one.

- a1p3.py can have a key that is a string upto the the length of the message. There are 52 letters to choose from for each letter in the string. Thus, there can be (52^(len(message)))! distinct keys (! is factorial). Yes, it is a really big number! How did I get this number? Lets say the message is "Dog". Then the key can be at max 3 letters (or 2 letters or 1 letter) so 52*52*52 or 52*52 or 52. So # of distinct keys is (52*52*52)*(52*52)*(52).

- Yes, the problem 2 Cipher is stronger than Caesar Cipher because it can have 52 distinct keys where Caesar Cipher can only have 26 distinct keys.

- Problem 1 and 2 are quite weak. Firstly, lets say the message is "DOG", there are 2 keys that can work, one would give "DOG" and other would give "dog". Secondly, the attacker would only need 52 tries (maximum - it could also be 26 like I said above) to break the code, which can take less than a minute to break. This issue is addressed with problem 3 where the key can be a string and there can be (52^(len(message)))! distinct keys. Even the frequncy analysis cannot break this.