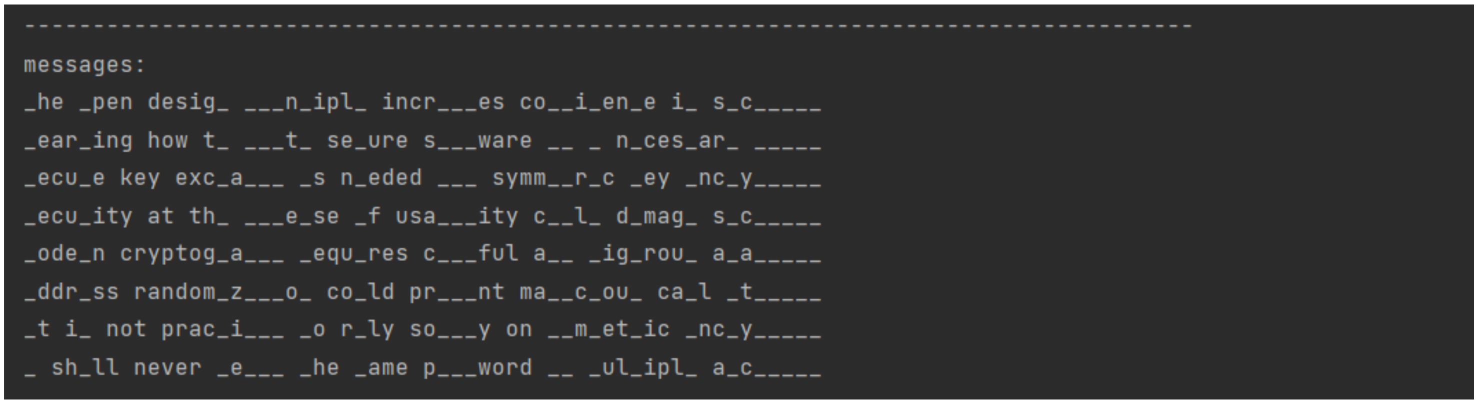
Message output by coding:



```
-------------------------------------------------------------------------------------
messages:
_he _pen desig_ ___n_ipl_ incr___es co__i_en_e i_ s_c_____
_ear_ing how t_ ___t_ se_ure s___ware __ _ n_ces_ar_ _____
_ecu_e key exc_a___ _s n_eded ___ symm__r_c _ey _nc_y_____
_ecu_ity at th_ ___e_se _f usa___ity c__l_ d_mag_ s_c_____
_ode_n cryptog_a___ _equ_res c___ful a__ _ig_rou_ a_a_____
_ddr_ss random_z___o_ co_ld pr___nt ma__c_ou_ ca_l _t_____
_t i_ not prac_i___ _o r_ly so___y on __m_et_ic _nc_y_____
_ sh_ll never _e___ _he _ame p___word __ _ul_ipl_ a_c_____
```

Messages Guessing:

the open design principle increases confidence in security

learning how to write secure software is a necessary

secure key extraction needed symmetric key encryption

security at the defense of usability could damage security

modern cryptography requires careful and rigorous analysis

address randomization could protect malicious call attacks

it is not practical to rely solely on symmetric encryption

i shall never reuse the name password of multiple accounts

Approach:

In this approach we are depending on the position of spaces.First of all,we will xor each cipher with others

counter example: c1 xor c2,c1 xor c3, c1 xor c4, c1 xor c5, c1 xor c6, c1 xor c7, c1 xor c8,and so on for rest of ciphers so we have a list of lists where each of the inside lists have the permutations for each cipher

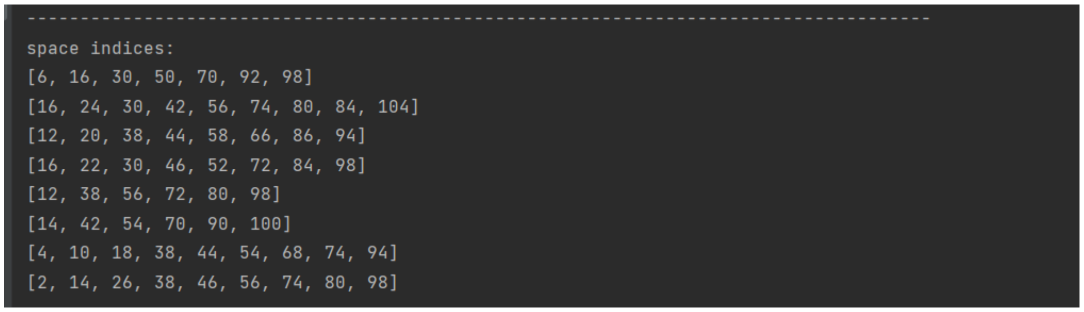
Since c1 xor c2= m1 xor m2

Then we have all the messages xored with each other,and from here we can say that by knowing that all messages are of letters of lowercase

According to [unicode - Xoring alphabet letters with space character - Stack Overflow]:

In ASCII, the uppercase letters are sequential binary numbers starting with A = 100.0001, while the lowercase letters are sequential binary numbers starting with a = 110.0001. In other words, a letter's case can be changed by flipping the second bit. Flipping the second bit is equivalent to a bitwise XOR with 010.0000. That happens to be the ASCII representation of the space character. It was certainly deliberate that uppercase and lowercase letters differ by just one bit. From this it follows that XORing any letter with the character represented by a 1 in that bit and a 0 in every other bit will flip its case. But it seems unlikely that this is the reason the space character was assigned to 010.0000. If it had made more sense for 010.0000 to be a period, or a dollar sign, or the digit 0, or any other character, it would still be just as easy to flip the case of a letter with XOR, using whatever that character was. As a conclusion, lower case letter Xor with space will give a capital letter, upper case letters start with A 41 hex.

By looping on all permutations of each cipher and comparing in the list of permutation each column of 2 digits (hex) by rest of permutations if found that the value is greater than 40 hex(upper case letter) or in all permutations value is equal to 00 then this means there is a space exists in this cipher where the permutations is compared I add this index to list of indices related to this cipher



```
-------------------------------------------------------------------------------------
space indices:
[6, 16, 30, 50, 70, 92, 98]
[16, 24, 30, 42, 56, 74, 80, 84, 104]
[12, 20, 38, 44, 58, 66, 86, 94]
[16, 22, 30, 46, 52, 72, 84, 98]
[12, 38, 56, 72, 80, 98]
[14, 42, 54, 70, 90, 100]
[4, 10, 18, 38, 44, 54, 68, 74, 94]
[2, 14, 26, 38, 46, 56, 74, 80, 98]
```

Where position 6 for example resembles 4th letter 01 23 45 67 so 4th column(letter) is space.

And then by looping on these indices on the main list

I check if the position is not in the space indices list then I add a dummy (_)

Else I check if the space is in first list then I put space in the first message and put in the rest of messages the letter after Xor again with space to return to lowercase and so on for rest of messages