**Applied Crypto**

**Many Time Pad**

We learned that one time pad is proved to be perfect security. What happens if we do many time pad, that is a key is to be used more than once. We need to find out. This is project is based on the famous problems from <http://crypto.stanford.edu/~dabo/cs255/hw_and_proj/hw1.html>.

# Problem 1

*describe one time pad and explain the reason it meets perfect security.*

* A key is used once for encryption and decryption
* Uses XOR

A close-up of a number

Description automatically generated

Perfect security because.

# Problem 2

*program to realize the one time Pad. It mimics a communication between Alice and Bob. Alice starts one program that asks Alice to enter a message on screen. The program then outputs the ciphertext on screen and saves the key generated randomly in a file. Bob then starts another program that reads the key and ciphertext and outputs the plaintext on screen. Assume Bob has a secret way to get the key file. The ciphertext can be input from screen or from a file.*

# Problem 3

*program to realize many time pad. This program hardcodes 10 plaintext messages you have chosen as a list, uses one randomly generated key to encrypt (xor) all the plaintext messages and saves the key and the ciphertexts in a file. Assume the key is longer enough.*

# Problem 4

Let us make an attack on the many time pad. Assume Eva has collected the 10 ciphertexts. Assume they are English, space is used, and no special characters.

1. 71fe1ace4389087266117cd7c98c4182851b3acff3b086e3f83f94d6eb05c4ba85d8e1fa14f11d1c3b568ff6cff5c09c5d67ef5c9c71b7eeb3d45a5154ab17b83e071ce9d8988adb4afedf46a840
2. 71fe1ace559a1e7266117cd7ce8745d7be2e74c3f0f68eeef57e8884e607debf81dfa0f012f95819681ae7f29fe4839b5175ef5e8760bef0b9d44b504eba12b22f5404f89dd085d550a48865a14f9b15a94dabe609ca2df2cccf210cefdb1af5389719795e1f0179cb77c5c456954d88f3
3. 72fe069c51c81a20775928c7879d4fd2a93c3acff3f69fe5fe2e9493a303d9ea98c4e5b60ae40a146058e7c787fbd09a1474e25dc865b5e6af865d4a40a61bfd384e06e0cfc1ccd356ff8853ac438905fa5fe3fd41cb3bbc8ac9
4. 67e543885b9a5b2267177084cf8453ccb8633ad7fdb39de5b13f8a93a304d6bf8bc4f4ef5def110b6f56a3e186e2c68c1470ef5c9c2ffbd6a291571e40ba1afd3b4b1fe0c4cbccc15df5dc07b043da01fa6ae4fd158f37b3c0cd
5. 71fe029a148c1236320d7192878a59cfbc3a6ec5e7f68befb13196d6ea1ec4ea81d9e3fe50ea0f196d02a2f7cfe2c29c5577e35d8630baf6ea80465b01aa1abc394f57a1f4ccccda59ff8846e44b8805bb5cabe608c231f2dec8364ae7d90ab4358c5c3a421b06
6. 6ef914ce5989152b321a769ad79c42c7be6f6ad2fab19de1fc339d84f04ad3a589dfa0ff09ab0c196f13e7e780b4c097556ded57c871fbeea393464a01aa0ab1381848cfd2d6898918efc046b00b8940bb08e3f313cb23b3dfd8645cfcd80ff82489
7. 71fe1ace4389087266117cd7c4865bd2b93b7fd2b5a58ce9f4308c9ff01e97ab82cbf2ef5dfc101d6a56b3fb8ab4d08b4167ef5c9c30b8f0ab97455b45e81efd364605e49ddb83df48eedc42b60c900fb14db4b229ca74b6c4d96442e1c34df8288f5c3a450a527ecc7c82865b8e
8. 71fe029a148c1437615978d7c58854dbec2c75cde5a39be5e37e9b97ef0697a285dfa0f01cff101d764983f29bf5
9. 71fe1ace50875b31730d6ad7cb8640c7ec3c73d4e1bf81e7b13796d6e518d8a4988ceff05dff101d2415a8fe9fe1d79a4623eb5e8430bfe3b3d442514faf40fd18420be0c8cb89924cf3cd5ee448950efd5cabe500c120f2d9d26440ebc34de029811977430b01748276d79012955cc6a65aebb9054becda5c9278
10. 71fe029a1483123c76597691878459cca9363ac4faf68ceffc2e8d82e61897b98fc5e5f809e20b0c7756b2e08aab83bc5560e257

Eva wants to use the information to decrypt the last message (target message).

71fe0680149d083b7c1e3996879a42d0a92e7780f6bf9fe8f42cd898e61cd2b8ccd9f3f35dff101d241da2eacff9cc8d5123fe5a897efbeda4974b

# **Problem 5** (optional)

*How many ciphertexts should we collect before we can do meaningful attack?*

Here are some observations that might be helpful to your attack thinking.

1. 0 Ꚛ x = x for any x in {0, 1}, 1 Ꚛ x = 1-x (in other words, it is the opposite)
2. kꚚk = 0 for any bit string.
3. From ASCII table, Space = 32 = 20Hex = 0010 0000
4. letter set A-Z: A = 65 = 0x41 = 0100 0001, … Z = 90 = 0x5A = 0101 1010
5. letter set a-z: a = 97 = 0x61 = 0110 1001, … z = 122 = 0x7A = 0111 1010.
6. space Ꚛ a =A, space Ꚛ A = a, and so on
7. XOR any two letters will give no letters as the first two bits of the xor will be 00
8. Any two ciphertext c1 = m1 Ꚛ k, c2 = m2 Ꚛ k, we have c1 Ꚛ c2 = m1 Ꚛ m2. This can tell us some characteristics, especially space about the plaintexts, which in turn reflected in the ciphertext.
9. c1 Ꚛ m1 = k
10. With given cyphertexts, we could figure out key structure partially or completely. We then decipher the last ciphertext.