6.857 Homework	**Problem Set 1**	# 1-2. One-time pad
Qian Long Joseph W Presbrey Mikhail Rudoy		February 19, 2014

1-2a)

Given two ciphertext C_1 and C_2 encoding using the same one-time-pad P, we can deduce the original messages M_1 and M_2 by xoring the ciphertexts, as follows.

$$\begin{aligned} M_1 \oplus P &= C_1 \\ M_2 \oplus P &= C_2 \\ M_1 \oplus M_2 &= C_1 \oplus C_2 \end{aligned}$$

To find the secret words we can first compute $C_1 \oplus C_2$ using the given values. Then, using the above relation, we simply have to find a pair of 8-character words M_1 and M_2 such that $M_1 \oplus M_2$ equals the computed value.

Given the assumption that the secret words are common 8-character English words, we take a list of common 8-character words from the internet and wrote a python script to look for the pair of words that, when xor-ed together, equals $C_1 \oplus C_2$. We optimize the performance of the search by first computing $M_1 \oplus C_1 \oplus C_2 = M_2$ for each M_1 in the list of words and storing the results in a hash table. Then we scan the word list again and simply check if the each word is in the hash table, taking $O(1)$ time for each check. If we find a word in the hashtable, then the pair M_1 and M_2 are found. Thus, this script runs in linear time.

The two words are security and networks.

Python code:

```
def ascii_word(string):
  output = []
  for char in string:
    output.append(ord(char))
  return output


def xor_word(array1, array2):
  output = []
  for i in xrange(len(array1)):
    output.append(array1[i] ^ array2[i])
  return output

def main():
  c1 = [0xe9, 0x3a, 0xe9, 0xc5, 0xfc, 0x73, 0x55, 0xd5]
  c2 = [0xf4, 0x3a, 0xfe, 0xc7, 0xe1, 0x68, 0x4a, 0xdf]
  result = xor_word(c1, c2)
  # str rep of ascii bytes
  resultstr = ''.join([str(x) for x in result])

  f = open('words.txt', 'r')
  words = []
  for line in f.readlines():
    words.extend(line.strip().split(' '))

  # find 2 words such that w1 ^ w2 == result
  # put result ^ w into hashtable => result ^ w1 = w2
  hashtable = {}
  for i in xrange(len(words)):
    m2 = ''.join([str(x) for x in xor_word(result, ascii_word(words[i]))])
    hashtable[m2] = words[i]

  for i in xrange(len(words)):
    ascii_str = ''.join([str(x) for x in ascii_word(words[i])])
    if ascii_str in hashtable:
      print "found match:"
      print hashtable[ascii_str]
      print words[i]
      break
  return

if __name__ == "__main__":
  main()
```

1-2b)