One-time Pad-Copy1

July 18, 2020

This notebook shows examples of One-time Pad encryption and partial decryption of the messages when the same key is used more than once for encryption.Before using XOR operation, we are going to convert plaintext messages to hex strings. This way, each character is represented by its ASCII code - a number from 0 to 255. In the hexadecimal system, any such number has two digits. We write these two hexadecimal digits instead of the initial character, like "6f" instead of letter "o", where ’f’ corresponds to hexadecimal number 15. We can also convert back from hex strings to the regular strings. These conversions are needed for demonstration purposes only: to avoid using unreadable/invisible characters in the strings we work with.

```
In [22]: # Converts string to hex
         def toHex(s):
             lst = []
             for ch in s:
                 hv = hex(ord(ch)).replace('0x', '')
                 if len(hv) == 1:
                     hv = '0'+hv
                 lst.append(hv)

             return reduce(lambda x,y:x+y, lst)

         print("toHex(\"Hello World\") = \"%s\"" % toHex("Hello World"))

         # Converts hex to string
         def toStr(s):
             return s and chr(int(s[:2], base=16)) + toStr(s[2:]) or ''

         print("toStr(\"736f6d65206d657373616765\") = \"%s\"" % toStr("736f6d65206d657373616765
```

```
toHex("Hello World") = "48656c6c6f20576f726c64"
toStr("736f6d65206d657373616765") = "some message"
```

```
In [23]: # Computes XOR of two messages s1 and s2.
         # s1 and s2 must have the same length.
         def Xor(s1, s2):
             res = ""
             for i in range(len(s1)):
                 res += format(int(s1[i], 16) ^ int(s2[i], 16), '01x')
             return res
```