Introduction to Computer Security

Lab 1 - Exploring Symmetric Key Cryptography

Report
of
Andreas Wilhelm

January 28, 2019

Part 1 - Warm It Up

```python
def xorfunc(txtCipher, txt):
    if len(txtCipher) != len(txt):
        raise NameError('The two strings hasn\'t equal lenght!')
    return bytes([a ^ b for (a, b) in zip(txtCipher, cycle(txt))])
    #cipher = XOR.new(txtCipher) #only 1 to 32 bytes long
    #return cipher.encrypt(txt)


if __name__ == '__main__':
    txtDarlin = "Darlin dont you go"
    txtHair = "and cut your hair!"


    print("txt:", txtDarlin)
    print("cipher:", txtHair)

    #print("\nencrypted: ", binascii.hexlify(xorfunc(txtHair,
   txtDarlin)))
    print("\nencrypted: ", '[{}]'.format(' ' .join(hex(x) for x in
   xorfunc(bytes(txtHair, 'utf-8'), bytes(txtDarlin, 'utf-8')))))
```

Listing 1: Code of task 1

Firstly implemted the task using pycrypto lib. However this lib has a limit of 32 Byte for the key, so it's not usable for longer strings. Therefore implemented the xorfunc by scratch. This functions will be used from the other tasks, as well. Realized the xorfunc with bytewise XOR with a for loop.

Part 2 - Implement OTP

```python
file = open(filePath, "rb").read()
print("file {} has {:d} characters" .format(filePath, len(file)))
print("filetxt: ", file)

cipher = os.urandom(len(file))
#print("cipherKeyHEX: ", binascii.hexlify(cipher))

txtEnc = xorfunc(cipher, file)
print("\nencryptedHEX: ", '[{}]'.format(' ' .join(hex(x) for x in
    txtEnc)))

fileEnc = open(filePathEnc, "wb").write(txtEnc)

txtDec = xorfunc(cipher, txtEnc)

print("\ndecrypted: ", txtDec.decode("utf-8"))
```

Listing 2: Code of task 2

For the One Time Pad (OTP) implemented reading and writing files in general. Additionally used the *os.urandom*-function to generate the cipher key with the same length as the file. For encrypting and decrypting the key with the file the xorfunc of listing 1 is used.

Part 3 - Encrypt an Image Using OTP

```python
def write_bmp_file(filename, input):
    byteEnc = open("./cp-logo.bmp", "rb").read(54)
    byteEnc += input[54:]
    open(filename, "wb").write(byteEnc)


def encryptBMPFile(picFilePath, picEncFilePath, cipher=os.urandom(
    fileLength)):
    file = open(picFilePath, "rb").read()
    print("file {} has {:d} characters".format(picFilePath, len(file
    )))

    txtEnc = xorfunc(cipher, file)
    # print("\nencryptedHEX: ", '[{}]'.format(' ' .join(hex(x) for x
    in txtEnc)))

    write_bmp_file(picEncFilePath, txtEnc)

    txtDec = xorfunc(cipher, txtEnc)

    open(picDec, "wb").write(txtDec)


def generateRndBMPFile(picEncFilePath, cipher=os.urandom(fileLength)
    ):
    write_bmp_file(picEncFilePath, cipher)


if __name__ == '__main__':
    encryptBMPFile(pic1, pic1Enc)
    encryptBMPFile(pic2, pic2Enc)
    generateRndBMPFile(pic3Enc)
```

Listing 3: Code of task 3