|  |  |  |
| --- | --- | --- |
|  |  |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |
| **Институт комплексной безопасности и специального приборостроения**  Кафедра КБ-4 «Интеллектуальные системы информационной безопасности»  Дисциплина «Кроссплатформенная среда исполнения программного обеспечения» | | |
|  |  |  |

**Отчет**

по проектной работе на тему

«Реализицая алгоритма шифрования RC4 на языке Python»

Выполнил:

Студент 2 курса

Группа ББСО-02-21

Шифр 21Б0613

Вандиловская Ксения Андреевна

Москва, 2022

1. **Задание проекта.**

Реализовать потоковый шифр RC4 на кроссплатформенном, интерпретируемом, языке программирования высокого уровня Python.

1. **Программный код с необходимыми комментариями и пояснениями.**

class Text(object)

В класс Text передается имя файла, над которым он будет проводить действия чтения/запись/преобразование данных.

Его методы:

def text\_to\_bytes(self)

def hex\_to\_bytes(self)   
  
def bytes\_to\_text(self, ByteList)   
  
def bytes\_to\_hex(self, ByteList)

В соответствие с порядком выполняют чтение данных из файла в виде dex; чтение данных из файла в виде hex; запись в файл в виде dex символов; запись в файл в виде hex символов.

Методы:

def encrypt(self, Key)   
  
def decrypt(self, Key)

В соответствие с порядком выполняют зашифровку и расшифровку данных.

Функция,

def crypt(PlainBytes, KeyBytes),

вызывается из двух последних методов класса Text. Она как раз и выполняет наложения ключа на текст, благодаря чему и происходит преобразование данных либо в виде читаемой информации либо в виде зашифрованного послания.

В функцию

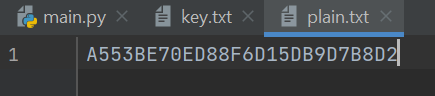
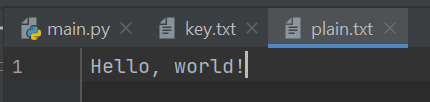
def main(name, Filename, Action, KeyName='key.txt', \*args)

Передаются аргументы программы: имя файла (Filename), действие (encrypt/decrypt), имя файла, в котором находится ключ. В нем создаются объекты типа Text и вызывается метод шифрования.

Код ниже отвечает за запуск программы и верное введенное количество аргументов:

if \_\_name\_\_ == '\_\_main\_\_':  
 if len (sys.argv) < 4:  
 print ('Usage: RC4.py filename encrypt/decrypt [keyfile]')  
 sys.exit (1)  
 else:  
 main(\*sys.argv)

**Пояснительная записка**

1. Реализация алгоритма шифрования RC4 на Python.
2. Вандиловская Ксения, ББСО-02-21.
3. В программу подается имя файла, который требуется зашифровать и имя файла, из которого будет браться ключ.
4. Для его реализации был написан собственный класс Text для обработки данных и импортирована библиотека sys для чтения аргументов в программу.
5. 
6. **Листинг кода.**

import sys  
  
class Text(object):  
  
 def \_\_init\_\_(self, filename):  
 self.filename = filename  
  
 def \_\_str\_\_(self):  
 return '%s' % self.filename  
  
 def text\_to\_bytes(self):  
  
 byteList = []  
  
 f = open(self.filename, 'r')  
 s = f.read()  
 f.close()  
  
 for byte in s:  
 byteList.append(ord(byte))  
  
 return byteList  
  
 def hex\_to\_bytes(self):  
  
 byteList = []  
 f = open(self.filename, 'r')  
  
 hexStr = f.read()  
 for i in range(0, len(hexStr), 2):  
 byte = hexStr[i:i + 2]  
 if '\n' in byte:  
 continue  
 byteList.append(int(byte, 16))  
  
 f.close()  
 return byteList  
  
 def bytes\_to\_text(self, ByteList):  
  
 s = ''  
 for byte in ByteList:  
 s += chr(byte)  
  
 f = open(self.filename, 'w')  
  
 f.write(s)  
  
 f.close()  
  
 def bytes\_to\_hex(self, ByteList):  
  
 f = open(self.filename, 'w')  
  
 for byte in ByteList:  
 hexStr = '0' + hex(byte)[2:]  
 f.write(hexStr[-2:].upper())  
  
 f.close()  
  
 def encrypt(self, Key):  
  
 PlainBytes = self.text\_to\_bytes()  
 KeyBytes = Key.text\_to\_bytes()  
 KeystreamBytes, CipherBytes = crypt(PlainBytes, KeyBytes)  
 Keystream = Text('keystream.txt')  
 Cipher = Text('plain.txt')  
 Keystream.bytes\_to\_hex(KeystreamBytes)  
 Cipher.bytes\_to\_hex(CipherBytes)  
  
 def decrypt(self, Key):  
  
 CipherBytes = self.hex\_to\_bytes()  
 KeyBytes = Key.text\_to\_bytes()  
 KeystreamBytes, PlainBytes = crypt(CipherBytes, KeyBytes)  
 Plain = Text('plain.txt')  
 Keystream = Text('keystream.txt')  
 Keystream.bytes\_to\_hex(KeystreamBytes)  
 Plain.bytes\_to\_text(PlainBytes)  
  
  
def crypt(PlainBytes, KeyBytes):  
 keystreamList = []  
 cipherList = []  
  
 keyLen = len(KeyBytes)  
 plainLen = len(PlainBytes)  
 S = list(range(256))  
  
 j = 0  
 for i in range(256):  
 j = (j + S[i] + KeyBytes[i % keyLen]) % 256  
 S[i], S[j] = S[j], S[i]  
  
 i = 0  
 j = 0  
 for m in range(plainLen):  
 i = (i + 1) % 256  
 j = (j + S[i]) % 256  
 S[i], S[j] = S[j], S[i]  
 k = S[(S[i] + S[j]) % 256]  
 keystreamList.append(k)  
 cipherList.append(k ^ PlainBytes[m])  
  
 return keystreamList, cipherList  
  
  
def main(name, Filename, Action, KeyName='key.txt', \*args):  
 try:  
 f = open(Filename, 'r')  
 f.close()  
 f = open(KeyName, 'r')  
 f.close()  
 except IOError:  
 print('File(s) do not exist.\nUsage: RC4.py filename encrypt/decrypt [keyfile]')  
 else:  
 if Action == 'encrypt':  
 Plain = Text(Filename)  
 Key = Text(KeyName)  
 Plain.encrypt(Key)  
 elif Action == 'decrypt':  
 Cipher = Text(Filename)  
 Key = Text(KeyName)  
 Cipher.decrypt(Key)  
 else:  
 print('Usage: RC4.py filename encrypt/decrypt [keyfile]')  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 if len (sys.argv) < 4:  
 print ('Usage: RC4.py filename encrypt/decrypt [keyfile]')  
 sys.exit (1)  
 else:  
 main(\*sys.argv)