Министерство образования Республики Беларусь

Учреждение образования

«Брестский Государственный технический университет»

Кафедра ИИТ

Лабораторная работа №1

По дисциплине «КМЗИ»

Тема: “Алгоритмы обмена ключами”

Выполнил:

Студент 3 курса

Группы ИИ-21

Романко Н.А.

Проверила:

Хацкевич А.С.

Брест 2023

1. Цель:Изучить теоретический материал методического указания. Ознакомится с алгоритмами обмена ключами. Получить вариант алгоритма обмена ключами у преподавателя.

Методы: EKE на основе ElGamal

Реализовать алгоритм обмена сеансовым ключом. Сеансовый ключ используется для шифрования/расшифрования информации.

1. Шифровать/расшифровывать информацию используя стандарт [RC4](file:///C:\Users\nikit\Downloads\Алгоритм%20RC4.doc)

Код программы клиента:

client = socket(

    AF\_INET, SOCK\_STREAM #Параметры подключения

)

client.connect(

    ('100.70.62.220', 7000)  #Адрес и сокет куда подключаться

)

password = b'mc0OEsmkyMW\_sAtNuyy1CnRfuErJmvllp-6\_Gv0bfMY='  #паорль

def send():

    message = input("Введите текст сообещния: ")

    client.send(message.encode('utf-8'))

def recive():

    print("Ожидание сообщения")

    msg = client.recv( 8192 ).decode('utf-8')

    print(f'SERVER MESSAGE:\n\t{msg}')

def rc4(key, plaintext):

    # Инициализация S-Box и массива ключа K

    S = list(range(256)

    K = [ord(key[i % len(key)]) for i in range(256)]

    # Перемешивание S-Box

    j = 0

    for i in range(256):

        j = (j + S[i] + K[i]) % 256

        S[i], S[j] = S[j], S[i]

    # Генерация ключевого потока и шифрование/дешифрование

    i, j = 0, 0

    ciphertext = []

    for char in plaintext:

        i = (i + 1) % 256

        j = (j + S[i]) % 256

        S[i], S[j] = S[j], S[i]

        key\_byte = S[(S[i] + S[j]) % 256]

        ciphertext.append(chr(ord(char) ^ key\_byte))

    return ''.join(ciphertext)

def main():

    sliceing = slice(2, -1)

    print("Waiting server response...")

    start\_message = "START"

    respond = ""

    client.send(start\_message.encode('utf-8'))

    while respond != "READY":

        respond = client.recv( 8192 ).decode('utf-8')

    print("Server responded")

    p = generate\_random\_prime()  #случайное число p

    g = generate\_random\_number() #случайное число g

    x = generate\_random\_number() #случайное число x

    while x == g:

        x = generate\_random\_number()

    r = (g ^ x) % p  #вычисление r

    #Открытым ключом являются r, g и p. И g, и p можно сделать общими для группы пользователей . Закрытым ключом является x.

    print("Generating key")

    r = str(r).encode('utf-8')

    client.send(r)

    f = Fernet(password)

    rec\_num = ""

    while rec\_num == "" :

        rec\_num = client.recv( 8192 )          #получение числа от сервера

    print("Random number receved")

    rec\_num = f.decrypt(rec\_num)

    print(rec\_num)

    print("Sending number to server")

    rec\_num = f.encrypt(rec\_num)      #отправка его же серверу

    client.send(rec\_num)

    d\_seans\_key = ""

    while d\_seans\_key == "":

        d\_seans\_key = client.recv( 8192 )     #получение зашифрованного сеансового ключа

    print("Seans key recieved, decrypting")

    seans\_key = f.decrypt(d\_seans\_key)

    seans\_key = seans\_key[sliceing]

    print("Seans key decrypted")

    print("Encrypting string, sending to server")

    f = Fernet(seans\_key)     #шифруем сообщение сеансовым ключем

    str1 = b"aboba send"

    encrypted\_string = f.encrypt(str1)

    client.send(encrypted\_string)

    str1 = "aboba send"   #люблю костыли)

    enc\_string1, enc\_string2 = "", ""

    while enc\_string1 == "" and enc\_string2 == "":

        enc\_string1 = client.recv( 8192 )

        enc\_string2 = client.recv( 8192 )

    print("String received, decoding")

    string1 = str(f.decrypt(enc\_string1))

    string1 = string1[sliceing]

    if string1 == str1:

        print("String coincided")    #расшифровка первой строки и проверка, совпали ои они

    string2 = str(f.decrypt(enc\_string2)) #расшифровка и повторная шифровка второй строки для отправки на сервер

    string2 = string2[sliceing].encode('utf-8')

    encrypted\_string2 = f.encrypt(string2)

    client.send(encrypted\_string2)

    print("Messages encrypted with RC4")

    seans\_key = str(seans\_key)[sliceing]

    print("Сеансовый ключ: ",seans\_key)

    while True:

        task = input("Выберете действие: \n\t 1-отправить сообщение\n\t 2-получить сообщение\n")

        match task:

            case "1":

                plaintext = input("Write message: ")

                encrypted = rc4(seans\_key, plaintext)

                print("Отправленное сообщение ", encrypted)

                client.send(encrypted.encode('utf-8'))

            case "2":

                encrypted = str(client.recv( 8192 ).decode('utf-8'))

                print("Полученное сообщение: ", encrypted)

                decrypted = rc4(seans\_key, encrypted)

                print("Расшифрованный текст:", decrypted)

            case "e":

                client.close()

                os.system("cls")

                break

            case \_:

                print("Нет такого лействия")

def is\_prime(n):

    if n <= 1:

        return False

    elif n <= 3:

        return True

    elif n % 2 == 0 or n % 3 == 0:

        return False

    i = 5

    while i \* i <= n:

        if n % i == 0 or n % (i + 2) == 0:

            return False

        i += 6

    return True

def generate\_random\_prime():

    while True:

        num = random.randint(10, 100)

        if is\_prime(num):

            return num

def generate\_random\_number():

    prime = generate\_random\_prime()

    num = random.randint(1, prime - 1)

    return num

Код сервера:

server = socket(

    AF\_INET, SOCK\_STREAM  #параметры подключения

)

server.bind(

    ('100.70.62.220', 7000)  #адресс и сокет сервера

)

server.listen(2)  #количество подключений

print("SERVER IS WORKING\n")

user, addr = server.accept()  #после подключения, выполнять далее

print(f"CONNECTED:\n\t{user}\n\t{addr}\n")

user.send('Connected successful!'.encode('utf-8'))  #отправить сообщение

password = b'mc0OEsmkyMW\_sAtNuyy1CnRfuErJmvllp-6\_Gv0bfMY='  #паорль

def send():

    message = input("Введите текст сообещния: ")

    user.send(message.encode('utf-8'))

def recive():

    print("Ожидание сообщения")

    msg = user.recv( 8192 ).decode('utf-8')

    print(f'SERVER MESSAGE:\n\t{msg}')

def rc4(key, plaintext):

    # Инициализация S-Box и массива ключа K

    S = list(range(256))

    K = [ord(key[i % len(key)]) for i in range(256)]

    # Перемешивание S-Box

    j = 0

    for i in range(256):

        j = (j + S[i] + K[i]) % 256

        S[i], S[j] = S[j], S[i]

    # Генерация ключевого потока и шифрование/дешифрование

    i = 0

    j = 0

    ciphertext = []

    for char in plaintext:

        i = (i + 1) % 256

        j = (j + S[i]) % 256

        S[i], S[j] = S[j], S[i]

        key\_byte = S[(S[i] + S[j]) % 256]

        ciphertext.append(chr(ord(char) ^ key\_byte))

    return ''.join(ciphertext)

def main():

    sliceing = slice(2, -1)

    print("Witing for client")

    msg = ""

    while msg !="START":

        msg = user.recv( 8192 ).decode('utf-8')

    message = "READY"

    user.send(message.encode('utf-8'))

    print("Clinet ready")

    f = Fernet(password)

    msg = ""

    while msg == "":

        msg = user.recv( 8192 )  #получение ключа от клиента

    print("Key recieved")

    print("Sending random encoded number")

    R = random.randint(1, 50)

    R1 = str(R).encode('utf-8')

    R\_enc = f.encrypt(R1)

    print("Sending number to client")

    user.send(R\_enc)

    R\_rec = ""

    while R\_rec == "" :

        R\_rec = user.recv( 8192 )

    print("Number received, decoding")

    R2 = f.decrypt(R\_rec)

    if int(R2) == R:

        print("Number coincided")

    print("generating seans key")

    seans\_key = Fernet.generate\_key()    #тоже самое что у клиента

    seans\_key1 = str(seans\_key)

    print("Encrypting seans key, sending to client")

    byte\_s\_key = seans\_key1.encode('utf-8')

    encrypted\_seans\_key = f.encrypt(byte\_s\_key)   #шифровка и отправка сеансового ключа

    user.send(encrypted\_seans\_key)

    encrypted\_string = ""

    while encrypted\_string == "":

        encrypted\_string = user.recv( 8192 )

    print("Encrypted string received")

    f = Fernet(seans\_key)

    decrypted\_string = str(f.decrypt(encrypted\_string))  #расшифровка строки клиента

    decrypted\_string = decrypted\_string[sliceing].encode('utf-8')

    print("String decoded")

    print("Encoding both strings, sending to client")

    string2 = b"aboba recive"

    encrypted\_string\_again = f.encrypt(decrypted\_string)  #шифруем строки сервера и клиента

    encrypted\_string2 = f.encrypt(string2)

    string2 = "aboba recive"

    user.send(encrypted\_string\_again)

    user.send(encrypted\_string2)

    enc\_ver\_string2 = ""  #получение строки от клиента для валидации

    while enc\_ver\_string2 == "":

        enc\_ver\_string2 = user.recv( 8192 )

    ver\_string2 = str(f.decrypt(enc\_ver\_string2))

    ver\_string2 = ver\_string2[sliceing]

    print("String received, decoding")

    if ver\_string2 == string2:

        print("String coincided")

    print("Messages encrypted with RC4")

    seans\_key = str(seans\_key)[sliceing]

    print("Сеансовый ключ: ",seans\_key)

    while True:

        task = input("Выберете действие: \n\t 1-отправить соо,щение\n\t 2-получить сообщение\n")

        match task:

            case "1":

                plaintext = input("Write message: ")

                encrypted = rc4(seans\_key, plaintext)

                print("Зашифрованное сообщение ", encrypted)

                user.send(encrypted.encode('utf-8'))

            case "2":

                encrypted = str(user.recv( 8192 ).decode('utf-8'))

                print("Полученное сообщение: ", encrypted)

                decrypted = rc4(seans\_key, encrypted)

                print("Расшифрованный текст:", decrypted)

            case "e":

                server.close()

                os.system("cls")

                break

            case \_:

                print("Нет такого лействия")

def is\_prime(n):

    if n <= 1:

        return False

    elif n <= 3:

        return True

    elif n % 2 == 0 or n % 3 == 0:

        return False

    i = 5

    while i \* i <= n:

        if n % i == 0 or n % (i + 2) == 0:

            return False

        i += 6

    return True

def generate\_random\_prime():

    while True:

        num = random.randint(10, 100)

        if is\_prime(num):

            return num

def generate\_random\_number():

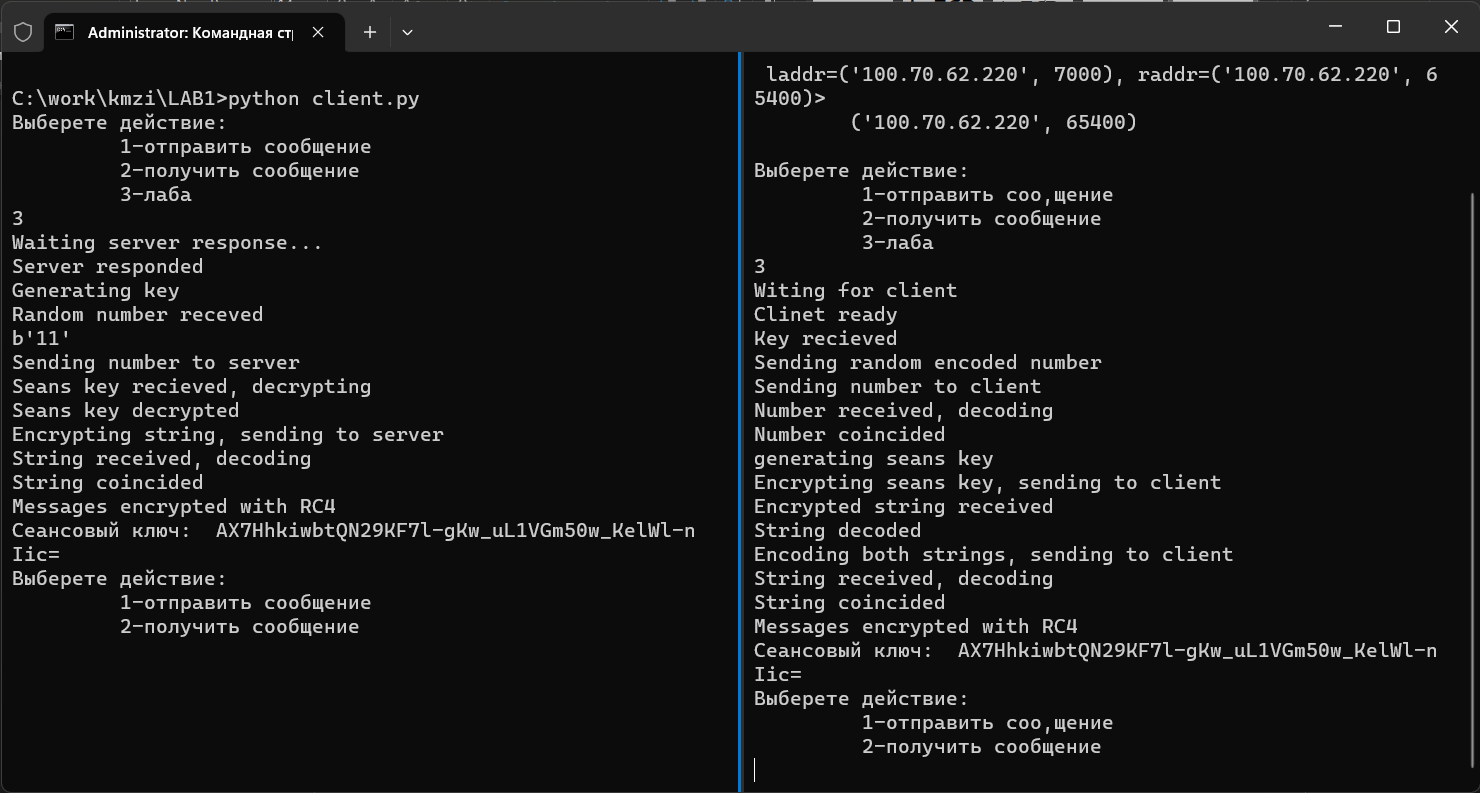
    prime = generate\_random\_prime()

    num = random.randint(1, prime - 1)

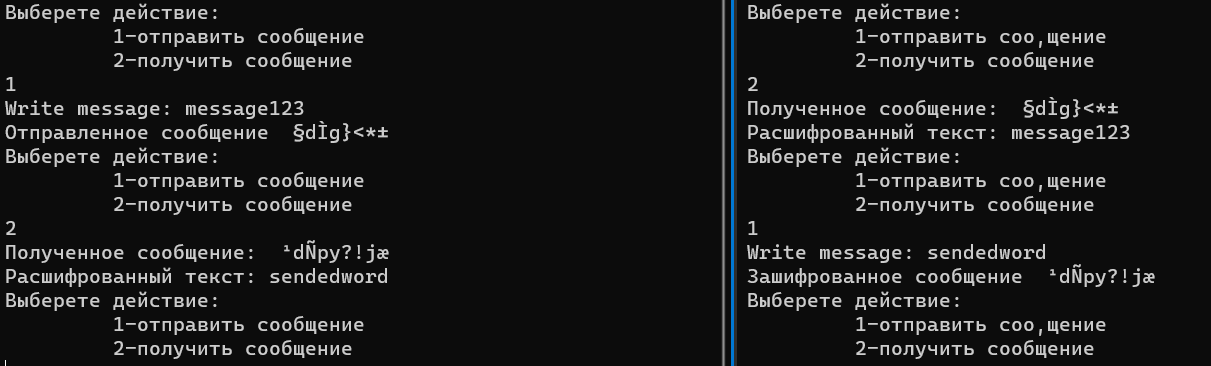
    return num

Вывод программы:

Обмен сеансовым ключом:



Обмен сообщениями зашифрованными RC4:



Вывод: изучил основные алгоритмы обмена ключами.