Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное  
учреждение высшего образования  
"Владимирский государственный университет имени Александра  
Григорьевича и Николая Григорьевича Столетовых"

(ВлГУ)

СМК 8/3-16

Срок хранения 6 месяцев

ВЛГУ.10.05.04.10.5.00 ПЗ

УТВЕРЖДАЮ

Заведующий кафедрой

доктор технических наук, профессор

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ М.Ю. Монахов

“\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 1

Дисциплина: Программирование

|  |  |  |
| --- | --- | --- |
| Исполнитель темы |  | ст. гр. ИСБ-120 Сухецкий Г.С |
|  | (подпись, дата) |  |
| Научный руководитель |  | доцент каф. ИЗИ, к.т.н. Монахов Ю.М. |
|  | (подпись, дата) |  |

Владимир 2022

СОДЕРЖАНИЕ

[**1 ФОРМУЛИРОВКА ЗАДАНИЯ 3**](#_Toc51786426)

[**2 ХОД ВЫПОЛНЕНИЯ РАБОТЫ 4**](#_Toc51786427)

[**3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ 5**](#_Toc51786429)

[**4 ВЫВОД 6**](#_Toc51786430)

# 1 Формулировка задания

1. Односторонняя аутентификация, основанная на метках времени:

После получения и расшифровывания данного сообщения участник В убеждается в том, что метка времени tA действительна и идентификатор В, указанный в сообщении, совпадает с его собственным. Предотвращение повторной передачи данного сообщения основывается на том, что без знания ключа невозможно изменить метку времени tA и идентификатор В.

2. Односторонняя аутентификация, основанная на использовании случайных чисел:

Участник В отправляет участнику А случайное число rB. Участник А шифрует сообщение, состоящее из полученного числа r, и идентификатора В, и отправляет зашифрованное сообщение участнику В. Участник В расшифровывает полученное сообщение и сравнивает случайное число, содержащееся в сообщении, с тем, которое он послал участнику А. Дополнительно он проверяет имя, указанное в сообщении.

3. Двусторонняя аутентификация, использующая случайные значения:

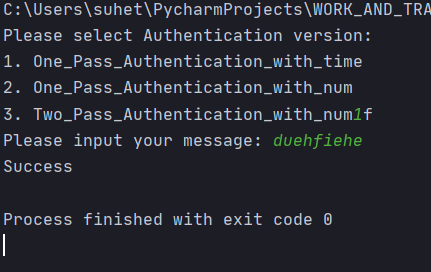
При получении сообщения (2) участник В выполняет те же проверки, что и в предыдущем протоколе, и дополнительно расшифровывает случайное число rА для включения его в сообщение (3) для участника А. Сообщение (3), полученное участником А, позволяет ему убедиться на основе проверки значений rАи rB, что он имеет дело именно с участником В.

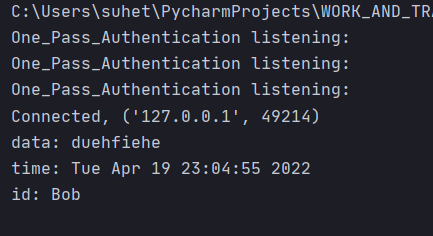
# **2.ХОД ВЫПОЛНЕНИЯ РАБОТЫ**

Для реализации данного задания был использован язык Python 3

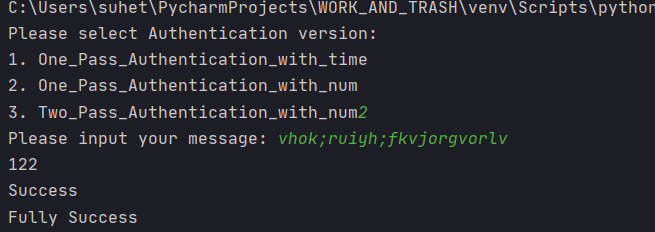
Программа.

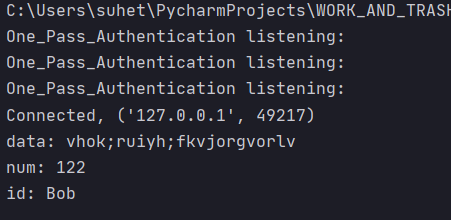
1. Односторонняя аутентификация, основанная на метках времени:



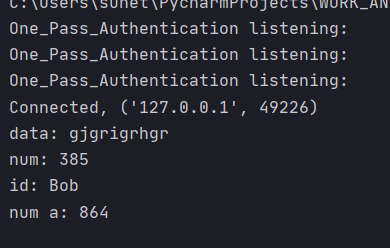


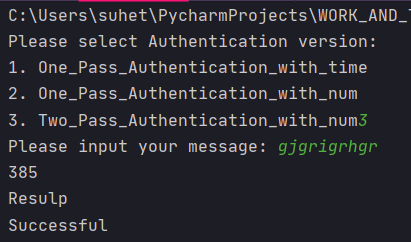
2. Односторонняя аутентификация, основанная на использовании случайных чисел:





3. Двусторонняя аутентификация, использующая случайные значения:





# **3 Программная реализация**

1. Односторонняя аутентификация, основанная на метках времени:

Server:

class One\_Pass\_Authentication:  
 def \_\_init\_\_(self):  
 self.socket = socket.socket()  
 self.socket.bind(('localhost'**,** 5000))  
 self.key = "This\_key\_for\_demo\_purposes\_only!"  
 self.id = "Bob"  
  
 def connect(self):  
 self.socket.listen(0)  
 print("One\_Pass\_Authentication listening:")  
 self.conn**,** self.addr = self.socket.accept()  
 print(f"Connected, {self.addr}")  
 self.recv()  
  
 def recv(self):  
 while True:  
 data = self.conn.recv(1024)  
 if not data:  
 break  
 if self.transform\_the\_data(data) == True:  
 results = "Success"  
 else:  
 results = "Error"  
 self.conn.send(results.encode("utf-8"))  
  
 def transform\_the\_data(self**,** dict):  
 self.key = self.key.encode("utf-8")  
 aes = pyaes.AESModeOfOperationCTR(self.key)  
 decrypted = aes.decrypt(dict)  
 data\_pikles = pickle.loads(decrypted)  
 print(f"data: {data\_pikles['data']}\n"  
 f"time: {data\_pikles['time']}\n"  
 f"id: {data\_pikles['id']}")  
 return self.check\_data(data\_pikles['data']**,**data\_pikles['time']**,**data\_pikles['id'])  
 def check\_data(self**,**data**,**timer\_check**,**id):  
 timer = time.asctime()  
 if timer\_check == timer and id == self.id:  
 return True  
 else:  
 return False

Client:

class One\_Pass\_Authentication\_with\_time:  
 def \_\_init\_\_(self):  
 self.data = ""  
 self.socket = socket.socket()  
 self.socket.connect(('localhost'**,** 5000))  
 self.key = "This\_key\_for\_demo\_purposes\_only!"  
 self.id = "Bob"  
  
  
 def input(self):  
 self.data = str(input("Please input your message: "))  
 self.transform\_the\_data()  
  
 def transform\_the\_data(self):  
 self.key = self.key.encode("utf-8")  
 aes = pyaes.AESModeOfOperationCTR(self.key)  
 timer = time.asctime()  
 dict = {"data": self.data**,** "time": timer**,** "id":self.id}  
 data\_pickle = pickle.dumps(dict)  
 ciphertext = aes.encrypt(data\_pickle)  
 self.send(ciphertext)  
  
  
 def send(self**,** data):  
 self.socket.send(data)  
 self.recv()  
  
 def recv(self):  
 data = self.socket.recv(1024)  
 print(data.decode("utf-8"))  
 self.socket.close()

2. Односторонняя аутентификация, основанная на использовании случайных чисел:

Server:

class One\_Pass\_Authentication\_with\_num:  
 def \_\_init\_\_(self):  
 self.socket = socket.socket()  
 self.socket.bind(('localhost'**,** 5001))  
 self.key = "This\_key\_for\_demo\_purposes\_only!"  
 self.id = "Bob"  
  
 def connect(self):  
 self.socket.listen(0)  
 print("One\_Pass\_Authentication listening:")  
 self.conn**,** self.addr = self.socket.accept()  
 print(f"Connected, {self.addr}")  
 self.recv()  
  
 def recv(self):  
 while True:  
 data = self.conn.recv(1024)  
 if not data:  
 break  
 try:  
 data = data.decode("utf-8")  
 self.num = secrets.randbelow(1000)  
 self.conn.send(str(self.num).encode("utf-8"))  
 self.recv()  
  
 except UnicodeDecodeError:  
 if self.transform\_the\_data(data) == True:  
 results = "Success"  
 else:  
 results = "Error"  
 self.conn.send(results.encode("utf-8"))  
  
 def transform\_the\_data(self**,** dict):  
 self.key = self.key.encode("utf-8")  
 aes = pyaes.AESModeOfOperationCTR(self.key)  
 decrypted = aes.decrypt(dict)  
 data\_pikles = pickle.loads(decrypted)  
 print(f"data: {data\_pikles['data']}\n"  
 f"num: {data\_pikles['num']}\n"  
 f"id: {data\_pikles['id']}")  
 return self.check\_data(data\_pikles['data']**,**data\_pikles['num']**,**data\_pikles['id'])  
 def check\_data(self**,**data**,**num\_check**,**id):  
 timer = time.asctime()  
 if num\_check == str(self.num) and id == self.id:  
 return True  
 else:  
 return False

Client:

class One\_Pass\_Authentication\_with\_num:  
 def \_\_init\_\_(self):  
 self.data = ""  
 self.socket = socket.socket()  
 self.socket.connect(('localhost'**,** 5001))  
 self.key = "This\_key\_for\_demo\_purposes\_only!"  
 self.id = "Bob"  
  
  
 def input(self):  
 self.data = str(input("Please input your message: "))  
 self.send("Start".encode("utf-8"))  
 self.recv()  
  
 def transform\_the\_data(self):  
 self.key = self.key.encode("utf-8")  
 aes = pyaes.AESModeOfOperationCTR(self.key)  
 timer = time.asctime()  
 dict = {"data": self.data**,** "num": self.num**,** "id":self.id}  
 data\_pickle = pickle.dumps(dict)  
 ciphertext = aes.encrypt(data\_pickle)  
 self.send(ciphertext)  
  
  
 def send(self**,** data):  
 self.socket.send(data)  
 self.recv()  
  
 def recv(self):  
 data = self.socket.recv(1024)  
 self.num = str(data.decode("utf-8"))  
 print(self.num)  
 if self.num == "Success":  
 print("Fully Success")  
 else:  
 self.transform\_the\_data()

3. Двусторонняя аутентификация, использующая случайные значения:

Server:

class Two\_Pass\_Authentication\_with\_num:  
 def \_\_init\_\_(self):  
 self.socket = socket.socket()  
 self.socket.bind(('localhost'**,** 5002))  
 self.key = "This\_key\_for\_demo\_purposes\_only!"  
 self.id = "Bob"  
  
 def connect(self):  
 self.socket.listen(0)  
 print("One\_Pass\_Authentication listening:")  
 self.conn**,** self.addr = self.socket.accept()  
 print(f"Connected, {self.addr}")  
 self.recv()  
  
 def recv(self):  
 while True:  
 data = self.conn.recv(1024)  
 if not data:  
 break  
 try:  
 data = data.decode("utf-8")  
 self.num = secrets.randbelow(1000)  
 self.conn.send(str(self.num).encode("utf-8"))  
 self.recv()  
  
 except UnicodeDecodeError:  
 if self.transform\_the\_data(data) == True:  
 results = "Success"  
 else:  
 results = "Error"  
 dict = {"result": results**,**"numA": self.numA}  
 results = pickle.dumps(dict)  
 self.conn.send(results)  
  
 def transform\_the\_data(self**,** dict):  
 self.key = self.key.encode("utf-8")  
 aes = pyaes.AESModeOfOperationCTR(self.key)  
 decrypted = aes.decrypt(dict)  
 data\_pikles = pickle.loads(decrypted)  
 self.numA = data\_pikles['numA']  
 print(f"data: {data\_pikles['data']}\n"  
 f"num: {data\_pikles['num']}\n"  
 f"id: {data\_pikles['id']}\n"  
 f"num a: {data\_pikles['numA']}")  
 return self.check\_data(data\_pikles['data']**,**data\_pikles['num']**,**data\_pikles['id'])  
 def check\_data(self**,**data**,**num\_check**,**id):  
 if num\_check == str(self.num) and id == self.id:  
 return True  
 else:  
 return False

Client:

class Two\_Pass\_Authentication\_with\_num:  
 def \_\_init\_\_(self):  
 self.data = ""  
 self.socket = socket.socket()  
 self.socket.connect(('localhost'**,** 5002))  
 self.key = "This\_key\_for\_demo\_purposes\_only!"  
 self.id = "Bob"  
  
  
 def input(self):  
 self.data = str(input("Please input your message: "))  
 self.send("Start".encode("utf-8"))  
 self.recv()  
  
 def transform\_the\_data(self):  
 self.key = self.key.encode("utf-8")  
 aes = pyaes.AESModeOfOperationCTR(self.key)  
 self.numA = secrets.randbelow(1000)  
 dict = {"data": self.data**,** "num": self.num**,**"numA":self.numA**,** "id":self.id}  
 data\_pickle = pickle.dumps(dict)  
 ciphertext = aes.encrypt(data\_pickle)  
 self.send(ciphertext)  
  
  
 def send(self**,** data):  
 self.socket.send(data)  
 self.recv()  
  
 def recv(self):  
 data = self.socket.recv(1024)  
 try:  
 data = data.decode("utf-8")  
 self.num = str(data)  
 print(self.num)  
 if self.num == "Success":  
 print("Fully Success")  
 else:  
 self.transform\_the\_data()  
 except UnicodeDecodeError:  
 print("Resulp")  
 data = pickle.loads(data)  
 if data["numA"] == self.numA:  
 print("Successful")  
 else:  
 print("Errors")

# **4 ВыВОд**

В ходе данной лабораторной работы был реализованы следующие алгоритмы аутентификации с использованием симметричного шифрования:

**1. Односторонняя аутентификация, основанная на метках времени:**

**2. Односторонняя аутентификация, основанная на использовании случайных чисел:**

**3. Двусторонняя аутентификация, использующая случайные значения:**