**Министерство науки и высшего образования РФ**

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Владимирский государственный университет имени Александра Григорьевича и Николая Григорьевича Столетовых»**

**(ВлГУ)**

УДК 004.8

Срок хранения 2 года

ВЛГУ.10.05.04.04.05.00 ПЗ

**ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ**

«**ЛР №2 - LAMPORT HASH CHAIN**»

Специальность 10.05.04 – «Информационно-аналитические системы безопасности»

|  |  |  |
| --- | --- | --- |
| Руководитель |  | к.т.н. доцент кафедры ИЗИ Д. В. Мишин |
|  |
| Исполнитель |  | студент группы ИСБ-119 Д. А. Журавлёв |
|  |  |  |  |

Владимир 2023 г.

**ЛР №2 - LAMPORT HASH CHAIN**

Программа должна моделировать процессы (на стороне клиента и сервера) регистрации и идентификации/аутентификации пользователя по протоколу LAMPORT HASH CHAIN.

Программа должна демонстрировать значения UserID, пароль (ключ), одноразовые пароли - хеш-значения (для различных итераций) и т.д. на этапах регистрации и идентификации/аутентификации.

**Алгоритм:**

**Процесс установки аутентификационных данных**

1. Клиент и сервер договариваются о числе N (из диапазона, скажем от 1.000 до 100.000) – эта величина несекретна и может быть просто hard-coded.
2. Клиент выбирает случайное число P большой размерности (128 бит и более):  
   — либо вычисляет его с помощью хеш-функции из пароля пользователя и имени сервера P=H(password||servername);  
   — либо генерирует его из доверенных источников случайных/псевдослучайных чисел и сохраняет на своей стороне (недостатком является появляющаяся при этом «привязка» к клиентской станции или необходимость использования токенов).
3. Клиент выполняет N раз рекурсивно криптостойкое хеширование H над числом P и передает получившийся результат H(H(H(...H(P)))) = HN(P) по любому каналу, защищенному от модификации, на сервер (*Внимание! Запись* HK(P) *в течение всей статьи будет использоваться для обозначения суперпозиции (K-кратного повторения вызова) хеш-функции, а не возведения ее результата в степень K*).
4. Сервер инициализирует счетчик A порядкового номера предстоящей аутентификации клиента числом 1.

**Процесс аутентификации**  
  
При каждой очередной необходимости проверки аутентичности пользователя:

1. Сервер высылает претенденту число A.
2. Клиент выполняет (N-A) раз хеширование над числом P и передает получившийся результат HN-A(P) серверу.
3. Сервер производит над числом HN-A(P) еще A раз хеширование и сверяет получившийся результат HA(HN-A(P)) с хранящимся у него HN(P).
4. При совпадении результата клиент считается успешно подтвердившим свою аутентичность, а сервер увеличивает счетчик успешных аутентификаций на 1: A=A+1, «сдвигаясь» тем самым по цепочке на одну позицию **влево**.

**Диаграмма последовательности:**

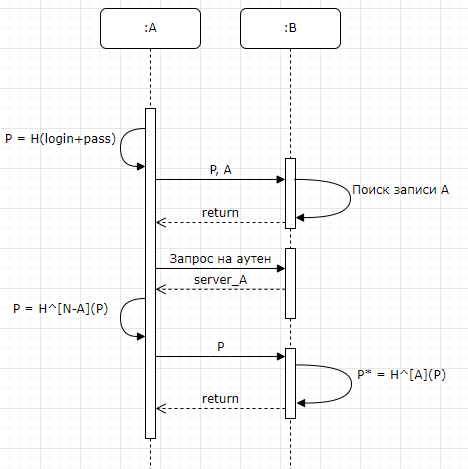


Рисунок 1 – Диаграмма последовательности

**Программа:**

Программа выполняет эмуляцию работы сервера и клиента при регистрации и аутентификации.

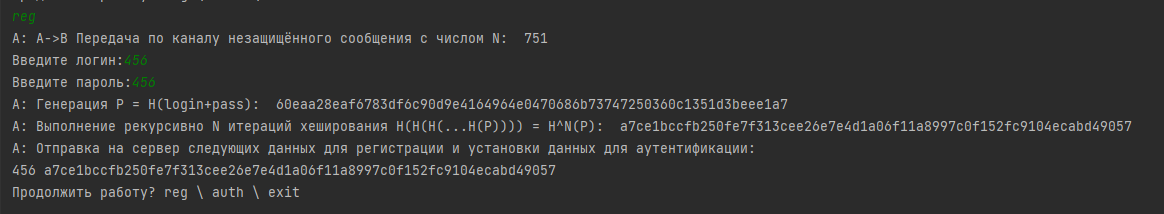


Рисунок 2 – Пример работы при регистрации

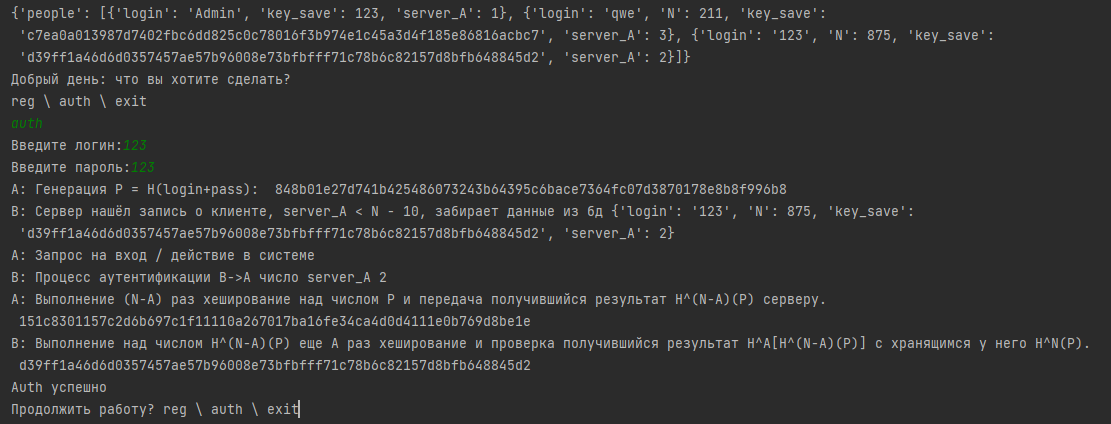
****

Рисунок 3 – Пример работы при аутентификации

Вывод:

В процессе работы был изучен LAMPORT HASH CHAIN. Реализована программа эмулирующая работу программы на стороне сервера и клиента.

**Приложение**

import os

import json

import hashlib

import random

def reg(users):

# начальный этап аутентификации

N = int(random.uniform(100, 1000))

print("А: A->B Передача по каналу незащищённого сообщения с числом N: ", N)

# генерация P хеш-функции из пароля пользователя и логина P=H(password login);

login\_ = input("Введите логин:")

pass\_ = input("Введите пароль:")

P = hashlib.pbkdf2\_hmac(

'sha256',

bytes((login\_ + pass\_).encode('utf-8')),

salt=bytes(''.encode('utf-8')),

iterations=100

)

print("А: Генерация P = H(login+pass): ", P.hex())

# Клиент выполняет N раз рекурсивно криптостойкое хеширование H над числом P

# и передает получившийся результат

# H(H(H(...H(P)))) = HN(P) по любому каналу,

# защищенному от модификации, на сервер

key\_save = P

for i in range(N):

key\_save = hashlib.pbkdf2\_hmac(

'sha256',

key\_save,

salt=bytes(''.encode('utf-8')),

iterations=100

)

print("А: Выполнение рекурсивно N итераций хеширования H(H(H(...H(P)))) = H^N(P): ", key\_save.hex())

# Сервер инициализирует счетчик A порядкового номера

# предстоящей аутентификации клиента числом 1.

server\_A = 1

print("A: Отправка на сервер следующих данных для регистрации и установки данных для аутентификации:")

newUser = {'login': login\_, 'N':N, 'key\_save': key\_save.hex(), 'server\_A': server\_A}

print(newUser['login'], newUser['key\_save'])

users.append(newUser)

return users

def auth(users):

# начальный этап аутентификации

# генерация P хеш-функции из пароля пользователя и логина P=H(password login);

login\_ = input("Введите логин:")

pass\_ = input("Введите пароль:")

if not any(u['login'] == login\_ for u in users):

print('Ошибка аутентификации. Неверный логин!')

return users

for u in users:

if u['login'] == login\_:

P = hashlib.pbkdf2\_hmac(

'sha256',

bytes((login\_ + pass\_).encode('utf-8')),

salt=bytes(''.encode('utf-8')),

iterations=100

)

print("А: Генерация P = H(login+pass): ", P.hex())

if u['server\_A'] >= (u['N'] - 10):

N = int(random.uniform(100, 1000))

print("А: A->B Передача по каналу незащищённого сообщения с логином "

"и числом N: ", login\_, " : ", N)

print("B: Сервер нашёл запись о клиенте, но server\_A < N - 10 клиент: ", u, ', запрос новых данных')

# Клиент выполняет N раз рекурсивно криптостойкое хеширование H над числом P

# и передает получившийся результат

# H(H(H(...H(P)))) = HN(P) по любому каналу,

# защищенному от модификации, на сервер

key\_save = P

for i in range(N):

key\_save = hashlib.pbkdf2\_hmac(

'sha256',

key\_save,

salt=bytes(''.encode('utf-8')),

iterations=100

)

print("А: Выполнение рекурсивно N итераций "

"хеширования H(H(H(...H(P)))) = H^N(P): ", key\_save.hex())

# Сервер инициализирует счетчик A порядкового номера

# предстоящей аутентификации клиента числом 1.

server\_A = 1

key\_save = key\_save.hex()

# Завершение процесса установки аутентификационных данных

else:

print("B: Сервер нашёл запись о клиенте, server\_A < N - 10, забирает данные из бд", u)

# забираем с бд данные по пользователю для его аутентификации

N = u['N']

key\_save = u['key\_save']

server\_A = u['server\_A']

# Завершение процесса установки аутентификационных данных

# Процесс аутентификации

print("А: Запрос на вход / действие в системе")

print("B: Процесс аутентификации B->A число server\_A", server\_A)

# клиент

key = P

for i in range(N-server\_A):

key = hashlib.pbkdf2\_hmac(

'sha256',

key,

salt=bytes(''.encode('utf-8')),

iterations = 100

)

print("A: Выполнение (N-A) раз хеширование над числом P и передача получившийся результат H^(N-A)(P) серверу.", key.hex())

#сервер

server\_key = key

for i in range(server\_A):

server\_key = hashlib.pbkdf2\_hmac(

'sha256',

server\_key,

salt=bytes(''.encode('utf-8')),

iterations = 100

)

print("B: Выполнение над числом H^(N-A)(P) еще A раз хеширование и проверка получившийся результат H^A[H^(N-A)(P)] с хранящимся у него H^N(P).", server\_key.hex())

if key\_save == server\_key.hex():

print("Auth успешно")

server\_A+=1

u['key\_save'] = key\_save

u['server\_A'] = server\_A

u['N'] = N

else:

print("Мимо")

return users

file\_path = 'bd.json'

if not os.path.exists(file\_path):

with open(file\_path, "w+") as f:

data = {'people': [{'login': 'Admin', 'key\_save':123, 'server\_A':1}]}

json.dump(data, f, indent=2)

with open(file\_path) as f:

file\_content = f.read()

items = json.loads(file\_content)

print(items)

print('Добрый день: что вы хотите сделать?')

print('reg \\ auth \\ exit')

users = items['people']

while(True):

k = input()

if k == 'reg':

users = reg(users)

with open(file\_path, "w") as f:

json.dump({'people': users}, f, indent=2)

print('Продолжить работу? reg \\ auth \\ exit')

elif k == 'auth':

users = auth(users)

with open(file\_path, "w") as f:

json.dump({'people': users}, f, indent=2)

print('Продолжить работу? reg \\ auth \\ exit')

elif k == 'exit':

print('Пока :( ')

break

else:

print('Неверная команда: reg \\ auth \\ exit')